

NPS ARCHIVE
1967
VILLARAN, T.

DETERMINATION OF OPTIMAL COMPLIANCE AND
STIFFNESS MATRICES FROM EXPERIMENTAL DATA

CESAR F. VILLARAN TAPIA

DETERMINATION OF OPTIMAL COMPLIANCE AND STIFFNESS
MATRICES FROM EXPERIMENTAL DATA

by

Cesar F. Villaran Tapia
Lieutenant, Peruvian Navy



Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1967

ABSTRACT

A linear structure can be characterized by its compliance matrix C , which is 6×6 symmetrical and positive definite and which relates a force 6-vector \bar{F}_j to a displacement 6-vector \bar{D}_j by the relation $C\bar{F}_j = \bar{D}_j$. The inverse, S , of C , is called the stiffness matrix and satisfies $S\bar{D}_j = \bar{F}_j$. This thesis deals with the problem of finding optimal values of such matrices C and S from experimental determinations of a sufficient number of vector-pairs (\bar{F}_j, \bar{D}_j) which are presumed to contain random errors.

J. E. Brock has introduced this problem area, suggested several different criteria of optimality, and solved some of the corresponding specific problems. This thesis completes the solution to a previously unsolved specific problem of this group and contributes computationally convenient new solutions to another. Moreover, a computer program, originally written for the CDC 1604 has been rewritten, in FORTRAN IV Language, as two programs for the IBM System 360 computer, and the capability has been significantly augmented.

TABLE OF CONTENTS

Section	Page
1 INTRODUCTION	7
1.1 Source and Statement of the Physical Problem	7
1.2 Brief History of this Problem and Related Problems	9
2 MATHEMATICAL DEVELOPMENT OF THE PROBLEM	10
2.1 Physical Statement of Criterion One	10
2.2 Alternate Criteria	11
2.3 Five Theorems on the Trace of a Matrix	12
2.4 Derivation of the Matrix Equations which assure the Optimization	13
3 SOLUTION OF EQUATIONS FOR CRITERIA 1 AND 2	17
4 SOLUTION OF EQUATION FOR CRITERION 3	20
5 SOLUTION OF EQUATION FOR CRITERION 4	25
6 DIGITAL COMPUTER PROGRAMS	27
7 TESTING OF THEORY AND DIGITAL COMPUTER PROGRAMS	34
8 CONCLUSIONS AND RECOMMENDATIONS	38
BIBLIOGRAPHY	43
APPENDIX	
A INSTRUCTIONS FOR USE AND LISTING OF PROGRAMS	44
A-1 Preparation of Data Deck and Final Assembly of Programs	44
A-2 Listing of Program CECI 1	47
A-3 Listing of Program CECI 2	51
B SUBROUTINE PROGRAMS	55
B-1 Description of Subroutines	55
B-2 Subroutine Listing	58
C SAMPLE PROBLEM SOLVED BY COMPUTER	76
C-1 Sample Problem Using Program CECI 1 with Zero Percent and Ten Percent Maximum Random Errors	76

C-2	Sample Problem Using Program CECI 2 with Zero Percent and Ten Percent Maximum Random Errors	89
D	BRIEF TREATMENT OF RELATED PROBLEMS	98
D-1	Case where there are no Restrictions on C	98
D-2	Case where C is Orthogonal	99

ACKNOWLEDGEMENTS

The writer wishes to gratefully acknowledge the patience and encouragement of his professor and faculty advisor, Dr. John E. Brock, without whose guidance this thesis would not have been followed to its completion.

The writer also wants to thank Mrs. Thomas Miles for her patience and well done job in correcting the English grammar of the original manuscript.

SECTION 1

INTRODUCTION

1.1 Source and Statement of the Physical Problem.

The coordinate system used here is a right handed orthogonal triad, whose unit vectors are denoted by \bar{e}_1 , \bar{e}_2 , and \bar{e}_3 ; see figure 1.

A generalized force will be denoted by \bar{F} , where

$$\bar{F} = \{ F_1, F_2, F_3, F_4, F_5, F_6 \}$$

is a column matrix. Braces $\{ \}$ indicate that a column matrix is intended. The first three elements, F_1 , F_2 , and F_3 , are force components in the \bar{e}_1 , \bar{e}_2 , and \bar{e}_3 direction respectively, and the last three elements, F_4 , F_5 , and F_6 , are moment components about these axes.

Also let a generalized deflection be denoted by \bar{D} , where

$$\bar{D} = \{ D_1, D_2, D_3, D_4, D_5, D_6 \}$$

The first three elements, D_1 , D_2 , and D_3 , are displacements in the \bar{e}_1 , \bar{e}_2 , and \bar{e}_3 direction respectively, and the last three elements are rotation components about these axes.

It is now presumed that a force \bar{F} is applied to a (suitably constrained) elastic structure, the application being at point P, and the corresponding deflection \bar{D} is observed at P; see figure 1.

It is postulated that a linear relation holds between the elements of \bar{F} and \bar{D} , i.e., that there exists a matrix C such that

$$C\bar{F} = \bar{D}$$

It is further postulated that matrix C is symmetric and positive definite, and that it has an inverse matrix S ($S = C^{-1}$) which, likewise, is symmetric and positive definite.

Matrix C is called the compliance (or flexibility) matrix (of the structure) and its elements are called compliance or flexibility coefficients.

Matrix S is called the stiffness matrix, and its elements are called stiffness influence coefficients.

Note that matrices C and S depend not only on the structure, but also on the choice of point P .

There are basically two approaches for obtaining the compliance matrix C , or the stiffness matrix S . One employs elastic theory in an attempt to come up with the elements of C or S . Even with simple structures it may not be easy to write down the elements of these matrices. The theory must be stretched beyond certainty and one obtains only an approximation to the correct result.

The second approach, the use of which leads to this thesis problem, employs experimental data rather than elastic theory to come out with the best C or S . Because of inevitable random errors present in the data, it is impossible to obtain the matrix C or S , without using some mathematical procedure that is able to minimize the influence of these errors, and at the same time to assure that matrices C and S are symmetrical and positive definite. Furthermore the data must contain a sufficient number of independent pairs of vectors \bar{F} and \bar{D} .

The question of how many pairs of vectors are required is considered in section 7. However in practice, it is anticipated that many more pairs will be available than the theoretical minimum.

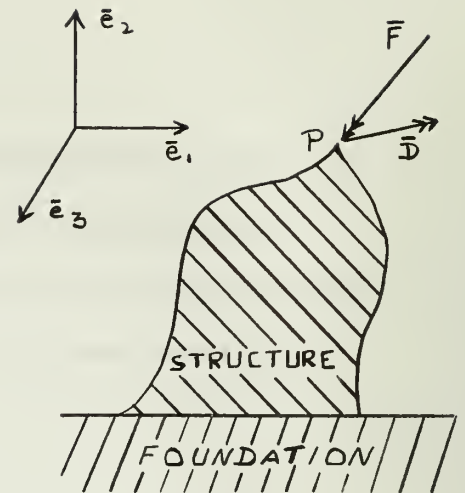


FIGURE 1

1.2 Brief History of this Problem and Related Problems.

Professor John E. Brock [1],¹ of the Mechanical Engineering Department at the Naval Postgraduate School, introduced this problem. He has defined different criteria of optimality and, corresponding to each such criterion, has established a matrix equation, the solution of which will minimize the corresponding errors and permit determining the desired symmetric and positive definite C or S matrix.

In the next chapter there will be given the mathematical descriptions of four criteria of optimality that Professor Brock has introduced in reference [1], and the developments leading to the matrix equations that which must be solved for an unknown square matrix.

A description and an explanation of how to use the computer programs which has been developed for solving these equations will be given later.

The data used to test the programs are not from an actual experiment. An explanation will be given of how sample data, containing random errors, are generated for test purposes.

In this thesis, matrix C has been restricted to be symmetrical and positive definite, as is required in the structural problem under consideration. Two other related situations will be considered in Appendix D.

In general the different criteria of optimality that will be given in the next chapter can be applied to all systems which are described by a matrix equation of the form $C\bar{F} = \bar{D}$, where C, \bar{F} , and \bar{D} , need only to be conformable for matrix operations.

¹Numbers in brackets refer to references listed in the Bibliography page 43.

SECTION 2

MATHEMATICAL DEVELOPMENT OF THE PROBLEM

2.1 Physical Statement of Criterion One.

It is assumed that there are n pairs of vector observations \bar{F} and \bar{D} .

As criterion one, it is desired to determine C so as to minimize the sum of the square of all scalar errors, viz., to minimize

$$\alpha_1 = \sum_{j=1}^n \sum_{i=1}^6 (\{C\bar{F}_j - \bar{D}_j\}_i)^2 \quad (1)$$

where $\{\}_i = i$ -th element of vector $\{\}$

Now let

$$U = [\bar{F}_1, \bar{F}_2, \bar{F}_3, \dots, \bar{F}_n]$$

$$V = [\bar{D}_1, \bar{D}_2, \bar{D}_3, \dots, \bar{D}_n]$$

These are $(6 \times n)$ matrices.

From equation (1)

$$\alpha_1 = \sum_{j=1}^n [(\{C\bar{F}_j - \bar{D}_j\}_1)^2 + (\{C\bar{F}_j - \bar{D}_j\}_2)^2 + \dots + (\{C\bar{F}_j - \bar{D}_j\}_6)^2] \quad (2)$$

Expanding the first term of equation (2)

$$\sum_{j=1}^n (\{C\bar{F}_j - \bar{D}_j\}_1)^2 = (\{C\bar{F}_1 - \bar{D}_1\}_1)^2 + (\{C\bar{F}_2 - \bar{D}_2\}_1)^2 + \dots + (\{C\bar{F}_n - \bar{D}_n\}_1)^2 \quad (3)$$

If all the terms in equation (2), are expanded as in equation (3), it is easy to see that these terms are nothing other than the diagonal terms of the matrix product: $(CU - V)^T(CU - V)$, and therefore it is possible to write

$$\alpha_1 = \text{tr} [(CU-V)^T (CU-V)] \quad (4)$$

where tr stands for the trace of a matrix and superscript T denotes the transposed of a matrix.

2.2 Alternate Criteria.

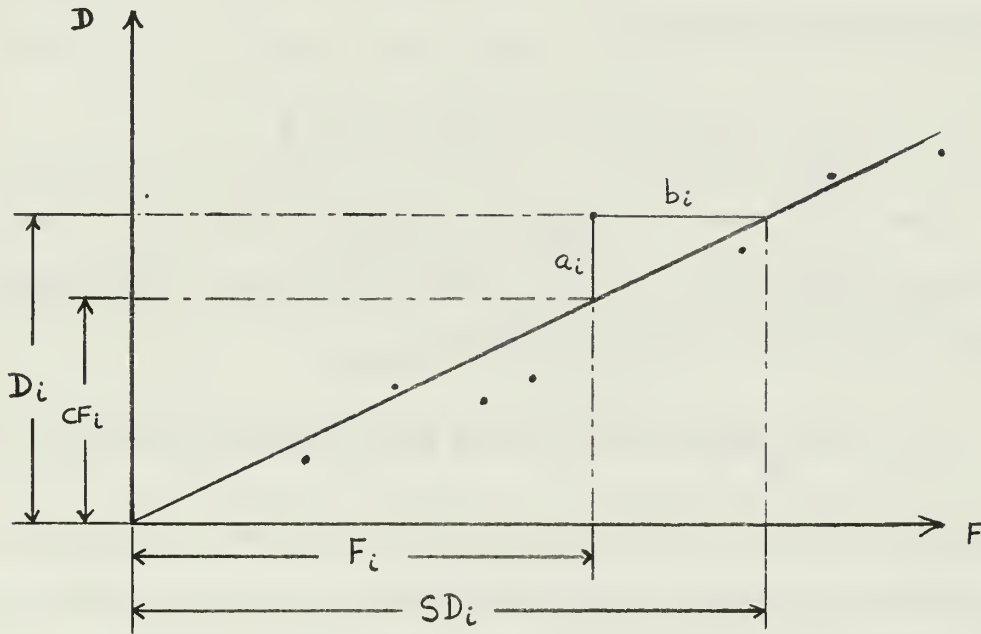


FIGURE 2

As may be seen from Figure 2, in a one dimensional case, criterion one requires minimizing the sum of the square of the vertical errors a_i . It is natural, therefore to consider, as criterion two, minimizing the sum of the squares of the horizontal errors, b_i . In the six dimensional case that this thesis is concerned with, this means minimizing

$$\alpha_2 = \sum_{j=1}^n \sum_{i=1}^6 (\{S\bar{D}_j - \bar{F}_j\}_i)^2 \quad (5)$$

Equation (5) is in the same form as equation (1), with the roles of \bar{F}_j and \bar{D}_j interchanged and using the stiffness matrix S instead of its

inverse, the compliance matrix C .

Using the same approach as before this sum can be written in the following way:

$$\alpha_2 = \text{tr} [(SV-U)^T(SV-U)] \quad (6)$$

It is possible too, to minimize the sum of the areas of the "error triangles", whose legs, in a one dimensional case, are a_i and b_i ; this is the same as minimizing

$$\alpha_3 = \text{tr} [(CU-V)^T(U-SV)] \quad (7)$$

Now, think also, of minimizing a weighted sum of the square of the horizontal and vertical errors $(\beta_1 \alpha_1 + \beta_2 \alpha_2)$. This criterion number four is the same as minimizing

$$\alpha_4 = \text{tr} [\beta_1 (CU-V)^T(CU-V) + \beta_2 (SV-U)^T(SV-U)] \quad (8)$$

where β_1 and β_2 are scalar quantities which may be specified so as to designate the relative importance of the two separate parts of which it is composed.

From equation (8), without loss of generality, it is possible to make $\beta_1 = 1$ and $\beta_2 = \beta$, and write equation (8) as

$$\alpha_4 = \text{tr} [(CU-V)^T(CU-V) + \beta (SV-U)^T(SV-U)] \quad (9)$$

Brock [1] also suggested a fifth criterion of optimality but since no progress has been made toward satisfying this criterion, it is not included here.

2.3 Five Theorems on the Trace of a Matrix.

In deriving the matrix equation corresponding to each criterion of

optimality, the following five simple theorems concerning the trace of a matrix will be used. The proofs are simple and are omitted.

THEOREM 1: If the matrix P and the matrix Q are conformable in either order, then each product is square and

$$\text{tr} (PQ) = \text{tr} (QP)$$

THEOREM 2: If the square matrices P and Q may be added, then

$$\text{tr} (P) + \text{tr} (Q) = \text{tr} (P + Q)$$

THEOREM 3: If P is a square matrix, then

$$\text{tr} (P) = \text{tr} (P^T)$$

THEOREM 4: If P is a matrix and x is a scalar, then

$$\left[\frac{\partial P}{\partial x} \right]^T = \frac{\partial P^T}{\partial x}$$

THEOREM 5: If P is a square matrix and x is a scalar, then

$$\frac{\partial \text{tr}(P)}{\partial x} = \text{tr} \left[\frac{\partial P}{\partial x} \right]$$

Note that theorem 1 permits one to write:

$$\text{tr} (ABCD) = \text{tr} (BCDA) = \text{tr} (CDAB), \text{ and so on.}$$

2.4 Derivation of the Matrix Equations which assure the Optimization.

In sections 2.1 and 2.2 four different criteria of optimality have been given. This section devotes attention to the derivation of a matrix equation corresponding to each criterion. Solving any such equation assures achieving the corresponding optimization.

a) Equation (4) is

$$\begin{aligned} \alpha_1 &= \text{tr} [(CU - V)^T (CU - V)] \\ &= \text{tr} (U^T C C U - V^T C U - U^T C V + V^T V) \end{aligned}$$

Note that instead of C^T , C has been written since C is symmetric.

Now let x be any given element of C, and let C^* denote $\frac{\partial C}{\partial x}$, then

$$\begin{aligned}
\frac{\partial \alpha_1}{\partial x} &= \text{tr} (U^T C^* C U + U^T C C^* U - V^T C^* U - U^T C^* V) \\
&= \text{tr} [(C U U^T + U U^T C - U V^T - V U^T)(C^*)] \\
&= \text{tr} [(C P + P C - Q)(C^*)]
\end{aligned}$$

where

$$P = U U^T, \text{ and } Q = U V^T + V U^T$$

are obviously symmetric.

Thus, if C satisfies the matrix equation

$$C P + P C = Q \quad (10)$$

then $\frac{\partial \alpha_1}{\partial x} = 0$, for all elements x of C and there is an extremum, which will be minimum, since α_1 is the sum of the squares of real numbers.

Because of the simple relationship between α_1 and α_2 , it is evident that a similar equation applies for criterion two.

This equation is

$$S R + R S = Q \quad (11)$$

where Q is as before and $R = V V^T$.

b) Equation (7) is

$$\alpha_3 = \text{tr} [(C U - V)^T (U - S V)]$$

The solution will be to determine the matrices C which make α_3 stationary; among these will be one which minimizes α_3 .

Knowing that $S = C^{-1}$, and both C and S are symmetric, after some manipulation

$$\alpha_3 = \text{tr} (U^T C U - V^T U - U^T V + V^T S V)$$

As before, let x be any given element of C , and let $C^* = \frac{\partial C}{\partial x}$
and $S^* = \frac{\partial S}{\partial x}$

Since $CS = I$, then

$$CS^* + C^*S = 0$$

and $S^* = -SC^*S$.

Thus

$$\begin{aligned} \frac{\partial \alpha_3}{\partial x} &= \text{tr} (U^T C^* U - V^T S C^* S V) \\ &= \text{tr} [(UU^T - SVV^T S)(C^*)] \end{aligned}$$

This will vanish and α_3 will be extremal if S satisfies the matrix equation

$$UU^T = SVV^T S$$

After post and pre-multiplying by C

$$CUU^T C = VV^T \quad (12)$$

c) Equation (9) is

$$\begin{aligned} \alpha_4 &= \text{tr} [(CU - V)^T (CU - V) + \beta (SV - U)^T (SV - U)] \\ &= \text{tr} [V^T V - U^T C V - V^T C U + U^T C C U + \\ &\quad \beta (V^T S S V - U^T S V - V^T S U + U^T U)] \end{aligned}$$

As before, let x be any given element of C .

Thus, using notation introduced previously, viz. :

$$P = UU^T, \quad R = VV^T, \quad \text{and} \quad Q = VU^T + UV^T$$

it may be seen that

$$\begin{aligned}
\frac{\partial \alpha_4}{\partial x} &= \text{tr} [-U^T C^* V - V^T C^* U + U^T C^* C U + U^T C C^* U + \\
&\quad \beta (V^T S^* S V + V^T S S^* V - U^T S^* V - V^T S^* U)] \\
&= \text{tr} [(-V U^T - U V^T + C U U^T + U U^T C)(C^*) + \\
&\quad \beta (S V V^T + V V^T S - V U^T - U V^T)(S^*)] \\
&= \text{tr} [(-Q + C P + P C)(C^*) + \beta (S R + R S - Q)(-S C^* S)] \\
&= \text{tr} [\{ (-Q + P C + C P) - \beta S (S R + R S - Q) S \} (C^*)]
\end{aligned}$$

which will vanish if

$$P C + C P - Q = \beta S (S R + R S - Q) S \quad (13)$$

If the right hand side of equation (13) is set equal to zero, the left hand side is precisely the same as equation (10), which corresponds to criterion number one. If the left hand side is set equal to zero, the right hand side is nothing other than equation (11), which corresponds to criterion number two.

SECTION 3

SOLUTION OF EQUATIONS FOR CRITERIA 1 AND 2

Equations (10) and (11) that minimize criteria 1 and 2 respectively are of the same form, where the matrices C, S, P, R, and Q, are all symmetric. The solution by Brock [1], given here, is for equation (10), but is applicable to equation (11), with the roles of P and R interchanged. The solution in equation (10) will be for the unknown matrix C, where in equation (11) is for the unknown matrix S.

In the physical problem of primary interest, matrices C, P, and Q are all 6 x 6, but, for the sake of generality, it will be presumed that they are m x m.

In equation (10)

$$CP + PC = Q$$

there are really only $\bar{N} = \frac{1}{2} m (m + 1)$ unknown elements of C. These unknowns can be arranged in a column matrix according to the following procedure:

Use the notation $M_j = \frac{1}{2} (j - 1) j$.

Given i, j, let $t_{i,j} = M_{\max(i,j)} + \min(i,j)$.

For example, with i = 3, and j = 5, then

$$t_{3,5} = t_{5,3} = \frac{1}{2} (4)(5) + 3 = 13$$

and

$$t_{2,6} = t_{6,2} = \frac{1}{2} (5)(6) + 2 = 17$$

Table 1 gives values of $t_{i,j}$ for i, j = 1, 2, 3, ..., 10

Next, a column matrix W of \bar{N} elements is formed from the \bar{N} unknown elements c_{ij} of matrix C. Element c_{ij} of C becomes the $t_{i,j}$ th element of W; note that $t_{m,m} = \bar{N}$.

	i=1	2	3	4	5	6	7	8	9	10
j=1	1	2	4	7	11	16	22	29	37	46
2	2	3	5	8	12	17	23	30	38	47
3	4	5	6	9	13	18	24	31	39	48
4	7	8	9	10	14	19	25	32	40	49
5	11	12	13	14	15	20	26	33	41	50
6	16	17	18	19	20	21	27	34	42	51
7	22	23	24	25	26	27	28	35	43	52
8	29	30	31	32	33	34	35	36	44	53
9	37	38	39	40	41	42	43	44	45	54
10	46	47	48	49	50	51	52	53	54	55

TABLE 1. Values of $t_{i,j}$

$$W = \{w_1, \dots, w_N\} = \{c_{11}, \dots, c_{mm}\}$$

The column matrix Z of \bar{N} elements is formed in a similar way from the known elements $q_{i,j}$ of symmetric matrix Q .

$$Z = \{z_1, z_2, z_3, \dots, z_N\}$$

where $z_{t_{i,j}} = q_{i,j} / (1 + \delta_{i,j})$

and $\delta_{i,j}$ is the Kronecker's delta.

Finally the coefficient matrix \bar{P} is formed as the sum of m ($\bar{N} \times \bar{N}$) matrices P_I , $I = 1, 2, \dots, m$, each of order $\bar{N} \times \bar{N}$, which themselves are formed from the $(m \times m)$ matrix $P = UU^T$, according to the following procedure:

P_I is formed from P by inserting rows and columns of zeros, an operation that is denoted by $(*)(K)(L)$ which means insert L rows of zeros after the K^{th} row in the original matrix P , and insert L columns of zeros after the K^{th} column in the original matrix P . The P_I is obtained from P by the operation:

- (a) $(*)(0)(M_I)$
- (b) $(*)(I)(I-1)$
- (c) $(*)(I+1)(I)$
- (d) $(*)(I+2)(I+1), \text{ etc.}$

until such insertions fall outside the range of the $\bar{N} \times \bar{N}$ size.

Then:

$$\bar{P} = \sum_{I=1}^m P_I$$

and solving the system

$$\bar{P}W = Z$$

gives the column matrix W, whose elements can then be rearranged to produce the desired matrix C.

Bickley and McNamee [4] have considered a more general equation of the form:

$$AZ + ZB = F$$

and they give four methods of solution, which they have named: The Big Matrix, the Irrational Solution, the Semi-Rational Solution, and A Rational Solution.

The method given by Larsen [5], and the method given by Brock [1] and reproduced here, appear to be additional distinct methods; both, however, probably are closely related to what Bickley and McNamee refer to as the "big matrix", in which $A = B$.

As the results obtained with the method given here is quite satisfactory, no further study has been done with the methods given in reference [4], but it might be of interest to one working with this problem to try suitably specializing Bickley and McNamee's methods.

SECTION 4

SOLUTION OF EQUATION FOR CRITERION 3

To solve equation (12), postmultiply by UU^T ; then

$$CUU^T CUU^T = VV^T UU^T \quad (14)$$

$$CUU^T = (VV^T UU^T)^{\frac{1}{2}} \quad (15)$$

In equation (14), UU^T and VV^T , both are symmetric, but the product $VV^T UU^T$ is non-symmetric.

Brock [6], has solved equation (15), using a highly sophisticated computer library subroutine, based on a Laguerre-Hessenberg algorithm, to find eigenvalues of $VV^T UU^T$, which are then used in a method described by Frazer, Duncan, and Collar [3], in order to find $CUU^T = (VV^T UU^T)^{\frac{1}{2}}$, from which C may easily be obtained. This procedure is available in a computer program for the FORTRAN 63 language to be used in the CDC 1604 computer.

Owing to the difficulties in converting this program to FORTRAN IV language used in the IBM System 360 computer, recently installed at the Naval Postgraduate School, the following two other procedures were developed and have successfully been applied to solve equation (12).

a) Iterative procedure for finding the square root of a non-symmetric matrix.

Pulay [7], has given a method that determines the principal square root of a positive definite symmetric matrix using an iteration procedure.

Using his idea with some modification, a method for obtaining the square root of a positive definite non-symmetric matrix has been successfully used and applied to several examples.

In equation (15)

$$CUU^T = (VV^TUU^T)^{\frac{1}{2}}$$

$$\text{let } VV^TUU^T = A$$

$$\text{and } A^{\frac{1}{2}} = D + B \quad (16)$$

where D is a diagonal matrix, whose elements are the positive square roots of the diagonal elements of A, and B is a matrix formed from $\tilde{C}UU^T - D$, and \tilde{C} is an approximation of the matrix C that will minimize the criterion number 3.

The matrix \tilde{C} can be obtained from the equation

$$\tilde{C}U = V$$

$$\tilde{C}UU^T = VU^T$$

$$\text{and } \tilde{C} = VU^T(UU^T)^{-1}$$

This is one way of finding the matrix \tilde{C} , suggested by analogy to Pulay's method. However, in the computer program used in this thesis to solve equation (15), the matrix \tilde{C} is taken to be the matrix C that satisfies criterion number 2. This is done in order to save computer time.

From equation (16)

$$(D + B)^2 = A$$

$$DB + BD = A - D^2 - B^2$$

This equation can not be solved for matrix B due to the term B^2 , but following Pulay's procedure it can be transformed into a simple iteration formula

$$DB_k + B_k D = A - D^2 - B_{k-1}^2$$

and the (i,j)-th element of B_k , can be obtained in a simple manner from the formula

$$(B_k)_{ij} = \frac{1}{D_i + D_j} \left[(A - D^2)_{ij} - (B_{k-1}^2)_{ij} \right]$$

To obtain $(B_k)_{ij}$, the iteration procedure must be carried out until the difference between two successive approximations of $(B_k)_{ij}$ become less than a specified allowable error, i.e.

$$(B_k)_{ij} - (B_{k-1})_{ij} = \Delta$$

where Δ in the actual program is 10^{-10} , and the $()_{ij}$ implies comparing element by element.

After matrix B has been obtained

$$C = (D + B)(UU^T)^{-1}$$

or

$$C = (VV^T UU^T)^{\frac{1}{2}} (UU^T)^{-1}$$

The procedure described above leads to output 3A in the actual computer program. Output 3B is the solution of basically the same equation, but instead of equation (12) being postmultiplied by UU^T , it is premultiplied resulting in

$$UU^T C = (UU^T VV^T)^{\frac{1}{2}}$$

and using the same procedure as before the square root of $UU^T VV^T$ can be obtained, and therefore the unknown matrix C.

Output 3C is simply the arithmetic average of 3A and 3B.

b) The second procedure that solves equation (12) uses some matrix algebra manipulation, and instead of working with a non-symmetric matrix, it requires taking the square roots of two different symmetric matrices. This procedure was called to the attention of the writer by Professor Brock.

Observe that starting with equation (12), $CUU^T C = VV^T$ and pre-multiplying by U^T and postmultiplying by U , the following matrix equation is obtained

$$U^T C U U^T C U = U^T V V^T U$$

and

$$U^T C U = (U^T V V^T U)^{\frac{1}{2}}$$

where $U^T V V^T U$ is a symmetric matrix, but as U and V have dimensions $(6 \times n)$, where n is the number of pair of vectors available, this formulation of the problem requires taking the square root to an $(n \times n)$ symmetric matrix, with the disadvantage of dealing with a big matrix, for example (1000×1000) .

Accordingly, return to equation (12)

$$C U U^T C = V V^T$$

Let $U U^T = A$, and without much loss of generality assume that the eigenvalues of A are distinct. In this problem A is a (6×6) matrix.

Then

$$A = M L^2 M^T$$

where M is the modal matrix of A

and L^2 is the spectral matrix of A .

This is true since A is symmetrical, L^2 is simply a diagonal matrix of the eigenvalues of A , and M is a square matrix whose columns are the corresponding eigenvectors of A . M is orthogonal so that $M^T = M^{-1}$.

Since L^2 is a diagonal matrix, it is easy to find its square root, L . Theoretically there are a number of different choices of L since either the positive or the negative square root of each eigenvalues can be chosen, but in this application it is convenient and satisfactory to use only the positive values.

Now

$$\begin{aligned} A &= M L^2 M^T = M L M^T \\ &= M L L^T M^T \\ &= E E^T \end{aligned}$$

where $E = M L$ is $(m \times m)$, i.e., 6×6 in the applied problem.

$$\text{Let } B = V V^T$$

and equation (12) now takes the form

$$CEE^T C = B$$

$$E^T CEE^T CE = E^T BE = D$$

where D is an (m x m) symmetric matrix. Thus,

$$E^T CE = D^{\frac{1}{2}}$$

Like matrix A, D can be written in terms of its modal and spectral matrices

$$D = m l_m^2 T$$

$$D^{\frac{1}{2}} = m l_m T$$

Therefore

$$E^T CE = m l_m T$$

Now

$$E^{-1} = L^{-1} M^{-1} = L^{-1} M^T$$

where L^{-1} is simply the diagonal matrix of the reciprocals of the corresponding elements of L,

Thus

$$C = M L^{-1} m l_m^T L^{-1} M^T$$

This procedure has been written in the FORTRAN IV language to be used in the IBM System 360 computer, and in the actual program is output 3D.

SECTION 5

SOLUTION OF EQUATION FOR CRITERION 4

The equation (13), that minimize the criterion 4, is nonlinear in the unknown matrix C ($= S^{-1}$), and an iteration method is used for its solution.

Given β , choose a convenient $\rho \leq \beta$. Now from equation (13), consider the iterative scheme indicated by

$$C_i P + P C_i - Q = \rho S_i (S_i R + R S_i - Q) S_i \quad (17)$$

where

$$S_i = C_{i-1}^{-1}$$

and
$$C_0 = \frac{\bar{C}_1 + \rho \bar{C}_2}{1 + \rho}$$

\bar{C}_1 is the matrix that satisfies criterion 1

\bar{C}_2 is the matrix that satisfies criterion 2

Now, having C_0 , S_1 is determined, from which the right hand side of equation (17) can be evaluated, and the following equation can be obtained:

$$C_i P + P C_i = Q^* \quad (18)$$

where

$$Q^* = Q + \rho S_i (S_i R + R S_i - Q) S_i$$

Equation (18) is of the same form as equation (10), and can be solved using the same procedure.

The C_i obtained from equation (18), is used again to obtain S_{i+1} , and a new C_{i+1} is obtained using the iterative procedure until two successive approximations of C_i become less than a specified allowable error, viz.

$$(C_{i+1})_{jk} - (C_i)_{jk} = \Delta$$

where Δ in the actual program is taken as 10^{-5} , and $()_{jk}$ implies comparing element by element.

When the convergence is achieved, ρ is increased and the iterative procedure is carried out again. This procedure is followed until finally convergence is obtained with $\rho = \beta$.

The reason for using values of ρ initially less than β is to assure and optimize convergence of the entire process. For example, in one study where $\beta = 0.8$, it was found convenient to use the sequence

$$\rho = 0.1, 0.2, 0.3, \dots, 0.8.$$

SECTION 6

DIGITAL COMPUTER PROGRAMS

Two main computer programs have been written in FORTRAN IV Language to be used with the IBM System 360 computer. These programs are called CECI 1 and CECI 2. They make use of several subroutines and to assure accuracy the main programs and the subroutines employ double precision arithmetic.

In program CECI 1, the following subroutines are used:

ADDEM	AINVM
BIGCM	BIGSM
BLANCA	JACVAT
MAUSS3	OUTPU
PRINM	RECIP
REDEM	SQRERT
SQROOT	SQUAM
TRACM	TRANM
ULTMA	ULTMB

In the program CECI 2, the following subroutines are used:

ADDEM	AINVM
BIGCM	BIGSM
BLANCA	MAUSS3
OUTPUT	PRINM
REDEM	SQUAM
TRACM	TRANM
ULTMA	ULTMB

Both programs make use of a number of functions from the monitor library tape which are not listed here.

Program CECI 1 solves the equations that satisfy criteria numbers 1, 2, and 3. The program gives one output for each of criteria 1 and 2, and four outputs for criterion number 3. For each of these outputs the program prints the compliance matrix C, and for each such matrix, computes and prints out three criteria numbers, as follows:

$$\text{CRITERION NUMBER ONE} = \text{tr} [(CU - V)^T (CU - V)]$$

$$\text{CRITERION NUMBER TWO} = \text{tr} [(SV - U)^T (SV - V)]$$

$$\text{CRITERION NUMBER THREE} = \text{tr} [(CU - V)^T (U - SV)]$$

Program CECI 2 solves the equation that satisfies criterion number 4, prints the compliance matrix C, and gives four criterion numbers; the first three are the same as in CECI 1 and the fourth is given by:

$$\begin{aligned} \text{CRITERION NUMBER FOUR} = & \text{tr} [(CU - V)^T (CU - V) + \beta (SV - U)^T \\ & (SV - U)] \end{aligned}$$

Most of the matrices are regarded as doubly subscripted arrays as follows:

If there are M rows and N columns, the array in the computer memory has dimension (M + 1, N); for example: if the matrix has 6 rows and 50 columns, the dimension of the array is (7 x 50). This extra row permits the identification of the matrix for debugging purposes and gives this information to those subroutines which use it for processing or monitoring purposes.

Let $e_{i,j}$ refer to the array and let $a_{i,j}$ refer to the matrix. Then $e_{1,1}$ is an integer identifying the array and the matrix. $e_{1,2}$ and $e_{1,3}$ are integers M and N respectively. For the remainder of the elements, $e_{i+1,j} = a_{i,j}$. The other matrices used in the main programs that have the dimensions (N,N), are used as scratch matrices. They are needed when calling some subroutines that have dummy subscripted variables, due

to the mode of storage (column-wise) used by the IBM System 360 computer. The array for an M-element column matrix has dimension $(M + 1)$, where the first element is the integer M.

The programs can be used in two different ways: (1) for classroom or more investigation purposes and (2) for use with data from an actual experiment. This is done by means of the card called MODE, where MODE = 1, 2, respectively.

If MODE = 1, the program will read the matrices C and U from the data cards, and generate matrix $V = CU$. For testing purposes random errors are introduced in matrices U and V. The explanation of how these errors are introduced will be given in the next section.

If MODE = 2, the programs will read the matrices U and V from the data cards. This data would be obtained from an actual experiment.

The explanation of how to use these programs, and how the data is punched in cards will be given in Appendix A.

Unless other FORMAT statements are called for, the only printed output is as follows:²

a) In main program CECI 1, with MODE = 1.

(1) MATRIX P FOLLOWS

MATRIX NUMBER 1, 6 ROWS, 6 COLUMNS

Listing of matrix P in FORMAT (1X, 8D12.4)

(2) MATRIX BE FOLLOWS

MATRIX NUMBER 2, 6 ROWS, NN COLUMNS

Listing of matrix BE in FORMAT (1X, 8D12.4)

² The nomenclature used in the outputs of the programs CECI 1 and CECI 2 is given in Appendix C; see pages 76 and 89.

- (3) MATRIX EB FOLLOWS
- MATRIX NUMBER 3, 6 ROWS, NN COLUMNS
- Listing of Matrix EB in FORMAT (1X, 8D12.4)
- (4) ERROR IN BLANCA = (number given)
- (5) MATRIX U FOLLOWS
- MATRIX NUMBER 4, 6 ROWS, NN COLUMNS
- Listing of matrix U in FORMAT (1X, 8D12.4)
- (6) MATRIX V FOLLOWS
- MATRIX NUMBER 5, 6 ROWS, NN COLUMNS
- Listing of matrix V in FORMAT (1X, 8D12.4)
- (7) ELAPSED TIME = (time given in sec.)
- (8) OUTPUT NO. 1, MINIMIZES V SQUARE ERROR
- DESIRED MATRIX IS
- MATRIX NUMBER 17, 6 ROWS, 6 COLUMNS
- Listing of desired compliance matrix in FORMAT (1X, 8D12.4)
- CRITERION NUMBER 1 EQUALS (value given)
- CRITERION NUMBER 2 EQUALS (value given)
- CRITERION NUMBER 3 EQUALS (value given)
- (9) ELAPSED TIME = (time given in sec.)
- (10) OUTPUT NO. 2, MINIMIZES U SQUARE ERROR
- DESIRED MATRIX IS
- MATRIX NUMBER 33, 6 ROWS, 6 COLUMNS
- Listing of desired compliance matrix in FORMAT (1X, 8D12.4)
- CRITERION NUMBER 1 EQUALS (value given)
- CRITERION NUMBER 2 EQUALS (value given)
- CRITERION NUMBER 3 EQUALS (value given)
- (11) ELAPSED TIME = (time given in sec.)

(12) OUTPUT NO. 3A, MINIMIZES TRIANGLE ERROR

DESIRED MATRIX IS

MATRIX NUMBER XXX, 6 ROWS, 6 COLUMNS

(XXX may have any large integer value, due to the iteration procedure used to obtain output 3A)

Listing of desired compliance matrix in FORMAT (1X, 8D12.4)

CRITERION NUMBER 1 EQUALS (value given)

CRITERION NUMBER 2 EQUALS (value given)

CRITERION NUMBER 3 EQUALS (value given)

(13) ELAPSED TIME = (time given in sec.)

(14) OUTPUT NO. 3B, MINIMIZES TRIANGLE ERROR

Same as number (12)

(15) ELAPSED TIME = (time given in sec.)

(16) OUTPUT NO. 3C, MINIMIZES TRIANGLE ERROR

Same as number (12)

(17) OUTPUT NO. 3D, MINIMIZES TRIANGLE ERROR

Same as number (12)

(18) ELAPSED TIME = (time given in sec.)

(19) TOTAL ELAPSED TIME = (time given in sec.)

b) In main program CECI 1, with MODE = 2.

(1) MATRIX U FOLLOWS

MATRIX NUMBER 1, 6 ROWS, NN COLUMNS

Listing of matrix U in FORMAT (1X, 8D12.4)

(2) MATRIX V FOLLOWS

MATRIX NUMBER 2, 6 ROWS, NN COLUMNS

Listing of matrix V in FORMAT (1X, 8D12.4)

(3) ELAPSED TIME = (time given in sec.)

- (4) OUTPUT NO. 1, MINIMIZES V SQUARE ERROR
DESIRED MATRIX IS

MATRIX NUMBER 14, 6 ROWS, 6 COLUMNS

Listing of desired compliance matrix in FORMAT (1X, 8D12.4)

CRITERION NUMBER 1 EQUALS (value given)

CRITERION NUMBER 2 EQUALS (value given)

CRITERION NUMBER 3 EQUALS (value given)

From here the printed out is the same as when MODE = 1.

- c) In main program CECI 2, with MODE = 1.

- (1) MATRIX P FOLLOWS

MATRIX NUMBER 1, 6 ROWS, 6 COLUMNS

Listing of matrix P in FORMAT (1X, 8D12.4)

- (2) MATRIX BE FOLLOWS

MATRIX NUMBER 2, 6 ROWS, NN COLUMNS

Listing of matrix BE in FORMAT (1X, 8D12.4)

- (3) MATRIX EB FOLLOWS

MATRIX NUMBER 3, 6 ROWS, NN COLUMNS

Listing of matrix EB in FORMAT (1X, 8D12.4)

- (4) ERROR IN BLANCA = (value given)

- (5) MATRIX U FOLLOWS

MATRIX NUMBER 4, 6 ROWS, NN COLUMNS

Listing of matrix U in FORMAT (1X, 8D12.4)

- (6) MATRIX V FOLLOWS

MATRIX NUMBER 5, 6 ROWS, NN COLUMNS

Listing of matrix V in FORMAT (1X, 8D12.4)

- (7) ELAPSED TIME = (time given in sec.)

- (8) ELAPSED TIME = (time given in sec.)

- (9) NUMBER OF ITERATIONS = (number given)

(10) OUTPUT NO. 4, MINIMIZES $V + BETA \times U$ SQUARE ERRORS

DESIRED MATRIX IS

MATRIX NUMBER XXX, 6 ROWS, 6 COLUMNS

Listing of desired compliance matrix in FORMAT (1X, 8D12.4)

BETA = (number given)

CRITERION NUMBER 1 EQUALS (value given)

CRITERION NUMBER 2 EQUALS (value given)

CRITERION NUMBER 3 EQUALS (value given)

CRITERION NUMBER 4 EQUALS (value given)

(11) ELAPSED TIME = (time given in sec.)

(12) TOTAL ELAPSED TIME = (time given in sec.)

d) In main program CECI 2, with MODE = 2.

(1) MATRIX U FOLLOWS

MATRIX NUMBER 1, 6 ROWS, NN COLUMNS

Listing of matrix U in FORMAT (1X, 8D12.4)

(2) MATRIX V FOLLOWS

MATRIX NUMBER 2, 6 ROWS, NN COLUMNS

Listing of matrix V in FORMAT (1X, 8D12.4)

From here the printed out is the same as when MODE = 1.

Actual printouts for both programs, when MODE = 1, are shown in Appendix C.

The listing of the main programs and subroutines will be given in Appendices A and B respectively.

SECTION 7

TESTING OF THEORY AND DIGITAL COMPUTER PROGRAMS

In this investigation the digital computer programs written to solve the matrix equations that satisfy the four criteria of optimality have not been tested using actual (mechanical) experimental data. The data have been generated by assuming a symmetric positive definite matrix C , and a set of vector forces $\{\bar{F}_j\}$ in the form of matrix U . From these two matrices the corresponding set of vector deflections $\{\bar{D}_j\}$ in the form of matrix V were obtained as $V = CU$.

To simulate experimental data, matrices U^* and V^* were formed in the following way:³

$$U^* = U + E(U)$$

$$V^* = V + E(V)$$

where $E(U)$ and $E(V)$ are matrices of errors.

Then matrices U^* and V^* were employed as if they were composed of experimental data, to be used in the programs to recover an approximation matrix C^* for C .

To form error matrices $E(U)$, and $E(V)$, a mathematical procedure was devised to perform some alterations in each element of the matrices, U and V , in a random manner with an upper limit to the absolute value of the percentage of error being introduced. This procedure has not been tested for randomness, but the method used to obtain the alterations suggests that the values of the elements in matrices $E(U)$ and $E(V)$ are indeed pretty good random errors.

³Note that the use of the asterisk,*, is different in this section from its use in section 2.

The random errors are generated by a subroutine called BLANCA. Because the first row of most of the matrices are used for identifying purposes, the subroutine makes use of the element(1,1), which is an identifying integer called NUMBER.

Subroutine BLANCA modifies a given matrix $A = [a_{ij}]$ by introducing random errors $E = [e_{ij}]$, resulting in a modified matrix $A^* = A + E$. One specifies the maximum percentage error to be applied.

First a number $\wp = 3.6327 + (\text{matrix identification NUMBER})$ is formed. \wp is increased for each element by adding a real constant (3.6327). It is not allowed to be greater than 9.999; this is done by subtracting a real constant (8.4153) each time \wp becomes greater than 9.999.

Next an exponent $\gamma' = (I/10 + J/25 + 0.62832)$ is formed. Next the number $\wp = \wp^{\gamma'}$ is formed. Next the number $\eta = (\text{fractional part of } \wp)$ is obtained. Finally the relative error is determined as $\eta \times r$, where r is the preselected maximum relative error. Then

$$e_{ij} = \pm \eta \times r \times a_{ij}$$

The algebraic sign is given by the first digit in the decimal expression of η . If this is even, the sign is positive; if this is odd, the sign is negative.

A special treatment is used if $a_{ij} = 0$, for which $e_{ij} = \pm \eta$, the sign being determined as before. This has obvious disadvantages if the average value of the elements in A is small, and it is suggested that future work make an appropriate modification to the present procedure.

In retrospect it appears that perhaps a more truly random variation could have been attained by using $\eta = (\text{fractional part of } 100 \wp)$, say. Furthermore, it would have been desirable to incorporate such randomizing influences as the date, or local time, or job sequence which

should become available internally to the computer, but which were not available at the time this work was done.

For convenience in comparing values of C^* with C , to see how good the recovered matrix C^* was, most of the compliance matrices C used were diagonal matrices. However, there is no loss in generality as may be seen from the following argument.

Starting with the matrix equation $CU = V$, with diagonal C , and pre-multiplying this equation by an orthogonal matrix M , it can be written as

$$MCM^T MU = MV \quad (19)$$

where the insertion of $M^T M$ in equation (19) between C and U does not alter the equality, due to the fact that $M^T M = I$.

Now let

$$MCM^T = \bar{C}$$

$$MU = \bar{U}$$

$$MV = \bar{V}$$

equation (19) can be written now as

$$\bar{C}\bar{U} = \bar{V} \quad (20)$$

Equation (20) has the same form as original matrix equation $CU = V$, but with the difference that starting with C (a diagonal matrix), the new symmetrical matrix \bar{C} is no longer a diagonal matrix, and instead all its elements are generally non-zero.

Thus a general problem $\bar{C}\bar{U} = \bar{V}$ could be written in the particular form $CU = V$, with diagonal C , if only the orthogonal matrix M , which diagonalizes C were known. $C = M^T \bar{C} M$, $U = M^T \bar{U}$, and $V = M^T \bar{V}$.

The particular validity of this argument was verified by two types of numerical experiment. In the first type, nondiagonal C 's were used, errors were introduced in U and $V = CU$, and the recovered matrix C^* was

successfully compared with the original C . In the second type, an orthogonal matrix M was generated, and matrices \bar{C} , \bar{U} , and \bar{V} were formed using M , diagonal C , U , and V . Then random errors were introduced in both sets of data (U , V and \bar{U} , \bar{V}) and C^* and \bar{C}^* were obtained. Comparison of $M^T C^* M$ with C was quite satisfactory.

From theory the minimum number of independent pairs of vectors \bar{F} and \bar{D} needed to solve the problem is 4. This is because in the compliance matrix C there are 21 unknowns, and 21 independent equations must be formed. Thus would require four pairs of 6-vectors \bar{F} and \bar{D} since the notion of $3\frac{1}{2}$ pairs is excluded. However, the particular methods employed in the programs seem to demand a larger number. The minimum number of pairs of vectors needed to obtain "good" results was six.

In one case, a problem with 24 pairs of vectors was solved twenty times, using a maximum random error of ten percent. The errors were introduced in a different manner for each of the solutions. From the twenty compliance matrices obtained in this way, an average compliance matrix was obtained. This matrix looks "very good", when compared element by element with the original compliance matrix. This indicates that if the value of n is so large as to overload computer memory or as to prove inconvenient for loading data, the U and V matrices may be conveniently partitioned, a number of "smaller problems" dealt with, and the answers averaged. One could regard the case described as forming a single problem with $n = 480$ which was actually treated as the average of the results of twenty problems, each having $n = 24$.

SECTION 8

CONCLUSIONS AND RECOMMENDATIONS

There does not seem to be any direct way of testing the validity of the theory and the computer programs which implement it other than by taking actual examples. These examples were constructed by assuming values for matrices C and U and generating matrix $V = CU$. Then, in most cases, errors were applied so as to get modified matrices U^* and V^* . Then, from these matrices, considered as input data to the program, the various procedures were used to recover an optimal value, C^* , which is then compared with the original C . No useful or convenient method has been found for assessing the accuracy of the recovery other than by comparing term by term, and, to simplify this task, in most cases C was assumed to be a diagonal matrix. It was shown in Section 7 that there is no loss in generality in doing this.

One of the things that might be expected to influence the ease and accuracy of the recovery is the relation between the eigenvectors of the matrix C . Tests were conducted, at the one extreme, with diagonal C having identical diagonal elements, and at the other extreme, with diagonal matrices having diagonal elements differing by a factor of 3125. In all, fairly extensive numerical work was investigated and various maximum percentage errors were introduced. The output produced as a result of these studies is much too voluminous to be reported in detail and it has not been found possible to reduce the volume of information in a satisfactory quantitative way. Accordingly, it is necessary to restrict remarks to those of a general nature which will be made in the following. In general, it may be concluded that the theory and the program have been proved to be accurate and effective, and in combination they provide a useful tool for analysing the data from certain types of

structural and mechanical experiments.

The following matrices have been used as compliance matrices C , from which the matrices C^* were recovered.

$$(1) \begin{bmatrix} 9. & 0 & 0 & 0 & 0 & 0 \\ 0 & 18. & 0 & 0 & 0 & 0 \\ 0 & 0 & 5. & 0 & 0 & 0 \\ 0 & 0 & 0 & 12. & 0 & 0 \\ 0 & 0 & 0 & 0 & 8. & 0 \\ 0 & 0 & 0 & 0 & 0 & 27. \end{bmatrix}$$

$$(2) \begin{bmatrix} 1. & 0 & 0 & 0 & 0 & 0 \\ 0 & 2. & 0 & 0 & 0 & 0 \\ 0 & 0 & 3. & 0 & 0 & 0 \\ 0 & 0 & 0 & 4. & 0 & 0 \\ 0 & 0 & 0 & 0 & 5. & 0 \\ 0 & 0 & 0 & 0 & 0 & 6. \end{bmatrix}$$

$$(3) \begin{bmatrix} 1. & 0 & 0 & 0 & 0 & 0 \\ 0 & 10. & 0 & 0 & 0 & 0 \\ 0 & 0 & 100. & 0 & 0 & 0 \\ 0 & 0 & 0 & 100. & 0 & 0 \\ 0 & 0 & 0 & 0 & 10. & 0 \\ 0 & 0 & 0 & 0 & 0 & 1. \end{bmatrix}$$

$$(4) \begin{bmatrix} 7. & 0 & 0 & 0 & 0 & 0 \\ 0 & 7. & 0 & 0 & 0 & 0 \\ 0 & 0 & 7. & 0 & 0 & 0 \\ 0 & 0 & 0 & 7. & 0 & 0 \\ 0 & 0 & 0 & 0 & 7. & 0 \\ 0 & 0 & 0 & 0 & 0 & 7. \end{bmatrix}$$

$$(5) \begin{bmatrix} 5. & 0 & 1. & 0 & 0 & 1. \\ 0 & 5. & 0 & 0 & 2. & 0 \\ 1. & 0 & 5. & 0 & 0 & 1. \\ 0 & 0 & 0 & 5. & 0 & 0 \\ 0 & 2. & 0 & 0 & 5. & 0 \\ 1. & 0 & 1. & 0 & 0 & 5. \end{bmatrix}$$

$$(6) \begin{bmatrix} 1. & 0 & 0 & 0 & 0 & 1. \\ 0 & 2. & 0 & 0 & 1. & 0 \\ 0 & 0 & 3. & 1. & 0 & 0 \\ 0 & 0 & 1. & 4. & 1. & 0 \\ 0 & 1. & 0 & 1. & 5. & 1. \\ 1. & 0 & 0 & 0 & 1. & 6. \end{bmatrix}$$

$$(7) \begin{bmatrix} 1. & 0 & 0 & 0 & 0 & 0 \\ 0 & 5. & 0 & 0 & 0 & 0 \\ 0 & 0 & 25. & 0 & 0 & 0 \\ 0 & 0 & 0 & 125. & 0 & 0 \\ 0 & 0 & 0 & 0 & 625. & 0 \\ 0 & 0 & 0 & 0 & 0 & 3125. \end{bmatrix}$$

OUTPUT No. 1 Gives the desired compliance matrix that satisfies criterion one. With this procedure the types of matrices numbered 1, 4, 5, and 6, have been recovered quite successfully using up to 35% maximum random

errors. Matrices numbered 3 and 7 were recovered with up to 5% maximum random errors, and matrix 2 was recovered with errors up to 25%. (Where no errors were introduced, each original was accurately recovered).

OUTPUT No. 2

Gives the desired compliance matrix that satisfies criterion two. All the matrices were recovered same as the originals when no errors were involved. Matrices numbered 1, 2, 3, 4, and 5 have been recovered quite successfully using up to 35% random errors, and matrices numbered 6 and 7 with up to 25% random errors.

OUTPUT Nos. 3A,3B,
3C

Gives the desired compliance matrix that satisfies criterion three, using the procedure that obtains the square root of a non-symmetric matrix. All the matrices were recovered same as the originals when no errors were involved. With matrices numbered 1 to 6 this procedure has been worked successfully with up to 35% random errors, and with matrix numbered 7 with up to 20% random error.

OUTPUT No. 3D

Gives the desired compliance matrix that satisfies criterion three, using the procedure that takes square roots of two different symmetric matrices. Matrix numbered 1 is the only one that has not been recovered same as the original when no errors were involved. In the light of other results obtained, this failure to recover matrix 1 to a very high degree of accuracy is unexpected and it has not been satisfactorily

explained. Matrices numbered 1 to 6 using up to 35% random error, have been recovered quite successfully, and matrix numbered 7 with up to 25%.

OUTPUT No. 4

Gives the desired compliance matrix that satisfies criterion four. Using any value for BETA, when no errors were involved, all the matrices were recovered same as the originals. When the error was increased, better results were obtained for small BETA's. The same remarks given for OUTPUT No. 1, applies to this output.

To have a better idea how these results look, see Appendix C, where the computer solutions for a particular problem with zero and ten percent random errors, are given.

The work presented here is just the beginning and not the last word in this new area; more work and investigation should be done. Some suggestions are given below:

1) Solution of equations that satisfy criteria 1, 2, and 4, can be investigated using the procedures given by Bickley and McNamee [4], or possible modifications to take advantage of the fact that matrices A and B are equal.

2) Investigate how large maximum random error percentages one can give to matrices U and V. Give to these matrices different maximum percentages of errors, e.g., to matrix U, 5%, and to matrix V, 15%.

3) There exists a fifth criterion given by Brock [1], that has not been investigated yet.

4) Devise a new iterative procedure for the equation that satisfies criterion four.

5) Devise a convenient criterion which measures the "overall" accuracy of recovery of a matrix C . At present, one merely compares element by element.

6) Employ advanced and sophisticated statistical theory for the analysis of errors and the accuracy of recovery. This would seem to imply constructing an adequate set of definition and the use of probability spaces, etc. The present analysis, in which general conclusions are attempted from specific cases, leaves much to be desired.

As things stand now, the engineering analyst has, in this theory and program implementation, a four edged tool which can be used for the analysis of experimental data. The decision of which criterion yields the "best" result is one which must be made on the basis of experience with the physical problem.

BIBLIOGRAPHY

1. Brock, John E. Second Progress Report, contract N161-26327 with John F. Wax, Inc., and United States Naval Academy, May 15, 1966.
2. Martin, Harold C. Introduction to Matrix Methods of Structural Analysis, McGraw-Hill, 1966.
3. Frazer, R. A., Duncan, W. J., and Collar, A. R. Elementary Matrices, Cambridge University Press and the Macmillan Co., Cambridge and New York, 1947.
4. Bickley, W. G., and McNamee, J. Matrix and other direct methods for the solution of systems of linear difference equations, Philosophical Trans. of the Royal Society of London, Series A., No. 1005, v. 252, 21 January 1960: 69-131.
5. Larson, H. J. Least squares estimation of the components of a symmetrical matrix. Techonometrics, v. 8, No. 2, May 1966: 360-362.
6. Brock, John E. Third Progress Report, contract N161-26327 with John F. Wax, Inc., and United States Naval Academy, July 15, 1966.
7. Pulay, Peter. An Iterative Method for the Determination of the Square Root of a Positive Definite Matrix. Zeit. d.Ang.Math. u. Mech. (ZAMM), v. 46, 1966: 151.

APPENDIX A

INSTRUCTIONS FOR USE AND LISTING OF PROGRAMS

A-1 Preparation of Data Deck and Final Assembly of Programs.

The preparation of the data deck is the same for programs CECI 1 and CECI 2, but has some alterations depending on whether $\text{MODE} = 1$ or $\text{MODE} = 2$.

In what follows the instructions of how to prepare the data deck is given for both types of modes.

The first card is bbbbbbb6NNbbblbb....., where b = blank, the 6 is the number of rows, NN is the number of columns, and 1 means first matrix to be read. When $\text{MODE} = 1$, NN = 06, is the number of columns in matrix C; when $\text{MODE} = 2$, NN varies from 06 to 50, and is the number of columns in matrix U. (Thus U is limited to a maximum of 50 columns).

Following this card, comes the elements of matrix C or U as the case may be, punched in five entries to a card, in FORMAT (5F14.6). Entries for the first row are thus entered until all elements for the row have been punched. Then, starting with a new card, entries for the second row are punched, and so on, until the entries for the sixth row have been entered. Each row starts with a new card.

Card bbbbbbb6NNbbb2bb....., follows. Then the entries for the matrix U (if $\text{MODE} = 1$) or matrix V (if $\text{MODE} = 2$) are punched in the same manner as before.

Example: Punch cards as in Figure A-1, for the following 6x6 diagonal matrix as C matrix.

$$C = \begin{bmatrix} 1. & 0 & 0 & 0 & 0 & 0 \\ 0 & 2. & 0 & 0 & 0 & 0 \\ 0 & 0 & 3. & 0 & 0 & 0 \\ 0 & 0 & 0 & 4. & 0 & 0 \\ 0 & 0 & 0 & 0 & 5. & 0 \\ 0 & 0 & 0 & 0 & 0 & 6. \end{bmatrix}$$

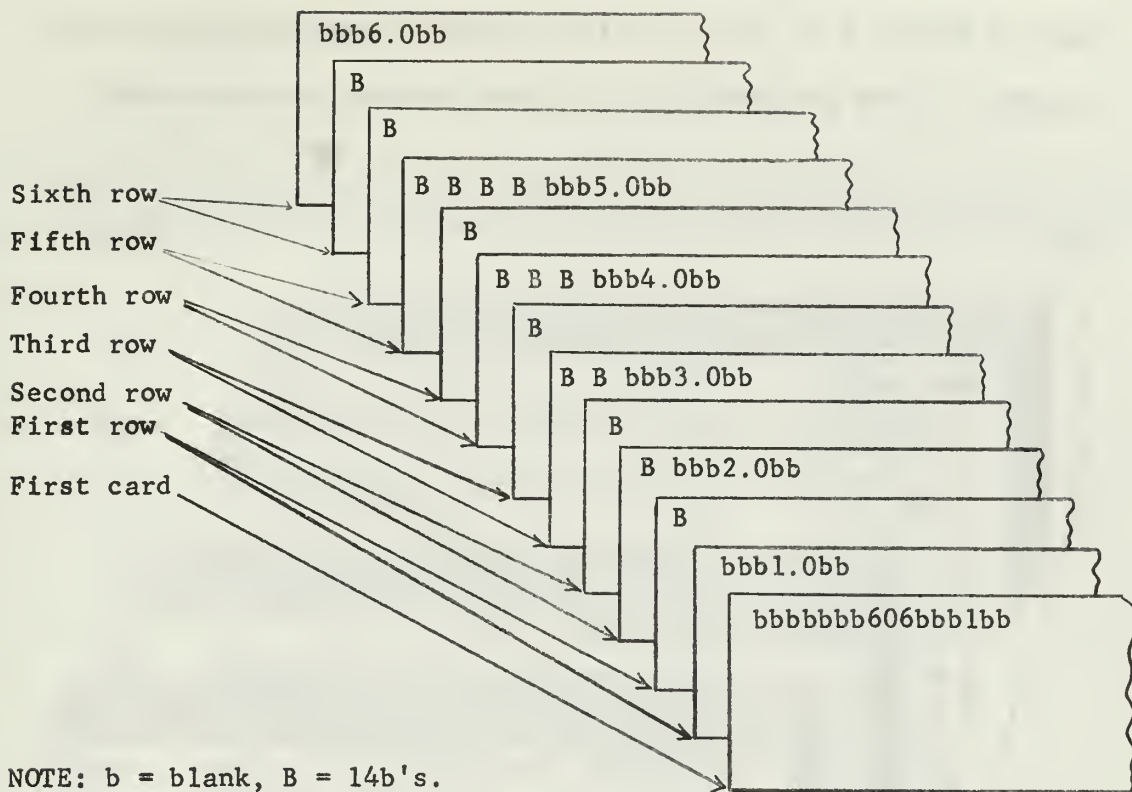


FIGURE A-1

Note that element C(1,1) is punched between fields 1 to 14, element C(2,2) between fields 15 to 28, and so on. Note too that each row has two cards because there are six elements for a row, and in each card only five consecutive elements can be punched.

If more than one set of data is used, the second set comes after the last card of the first set, starting again with the first card

bbbbbbbbNNbbb1bb.....

and so on.

In order to stop the program, the final card of the data deck is punched as follows bbbbbbb0b6bbb6bb.....

The complete program deck includes: control cards, main program, subroutine package, and data deck. The assembly of these components is

shown in Figure A-2, where the first control card is from the CECI 1 program. To use program CECI 2, change the word CECI 1 to CECI 2.

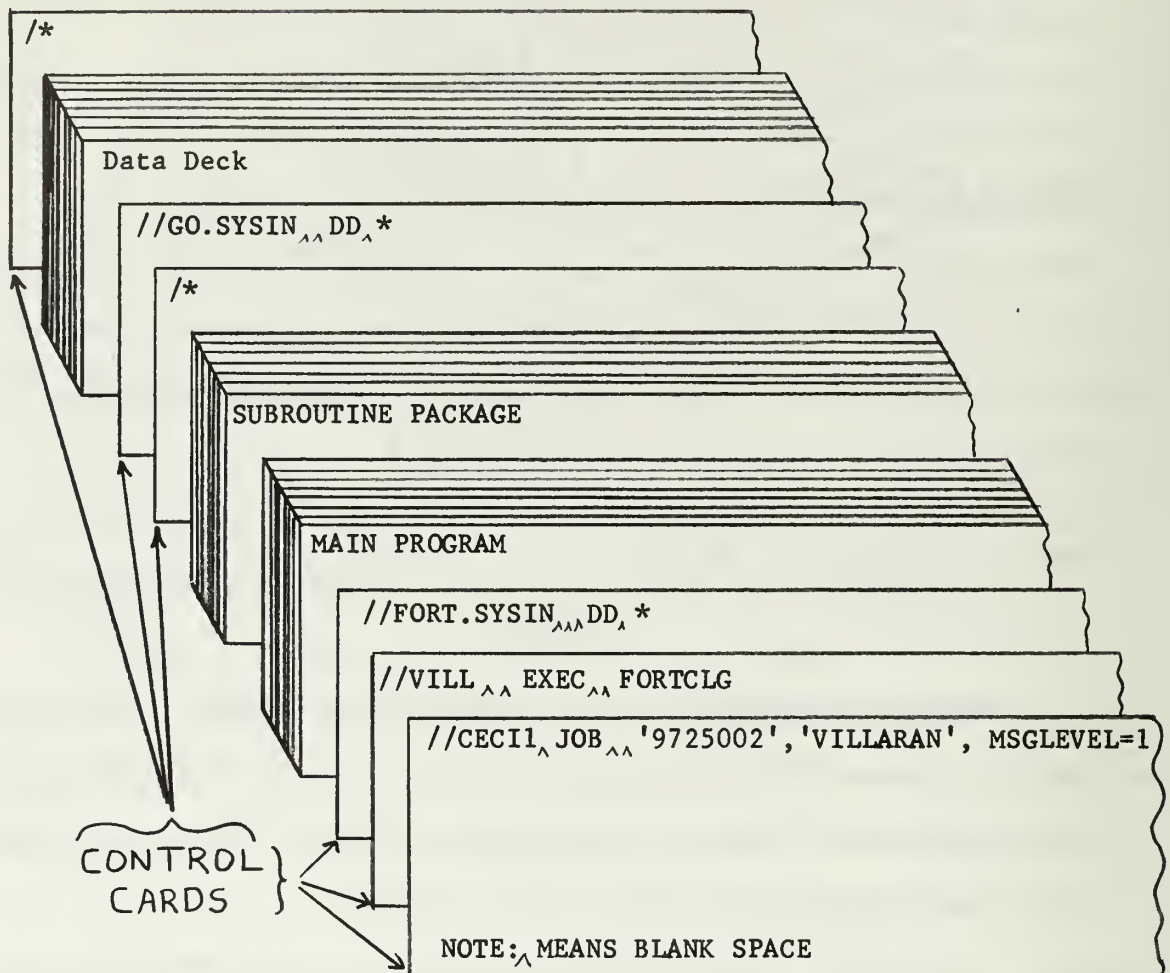


FIGURE A-2

A-2 Listing of Program CECI 1.

The cards of program CECI 1 are numbered from 0000 to 0159.

```

0000 IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
0001 DIMENSION T2(51,6),Z(21,21),X(21,21),AB(6,6),
0002 1Z2(6,6),BE(7,50),EB(7,50),BOOT(7,6),U(7,50),V(7,50),TR(51,6),
0003 2P(7,6),Y(7,6),PI(7,6),ROOT(7,6),W(22,21),S(22,21),COL(22),AA(22),
0004 3Q(7,6),QQ(7,6),GG(7,6)
0005 COMMON NUMBER
0006
0007 PROGRAM CECI 1 "SOLVES" THE OVERDETERMINED LINEAR SYSTEM XU=V, GIVEN
0008 WHERE X IS AN UNKNOWN SYMMETRIC (6,6) MATRIX AND U AND V ARE GIVEN
0009 RECTANGULAR MATRICES. THREE SOLUTIONS ARE GIVEN, WHICH "SOLVE" THE
0010 PROBLEM IN THE SENSE OF SATISFYING EACH OF THREE, ERROR CRITERIA.
0011 FOR DETAILS SEE NAVAL POSTGRADUATE SCHOOL THESIS, "DETERMINATION
0012 OF OPTIMAL COMPLIANCE AND STIFFNESS MATRICES FROM EXPERIMENTAL
0013 DATA", BY LT. C. F. VILLARAN, 1967. CARD NUMBER 0043 IS EITHER
0014 MODE=1 OR MODE=2. IF MODE=2, THE PROGRAM READS INPUT MATRICES U
0015 AND V FROM PUNCHED CARDS ACCORDING TO FORMAT (5F14.6). READING IS
0016 BY COMPLETE ROW, A NEW ROW STARTING ON A NEW CARD. CARDS FOR MATRIX
0017 IS AS FOLLOWS: 888888886NN8888 WHERE 8 DENOTES BLANK, NN IS THE
0018 TWO DIGIT NUMBER OF COLUMNS IN U AND V, NN NOT GREATER THAN 50, U
0019 AND M = 1 FOR MATRIX U, M = 2 FOR MATRIX V. CARDS FOR MATRICES U
0020 AND V IMMEDIATELY FOLLOW THE FINAL TWO CONTROL CARDS (/* AND
0021 //GO.SYSIN DD *) IN THE PROGRAM. IF THERE IS MORE THAN ONE SET OF
0022 INPUTS, THESE ARE PLACED IN ORDER FOLLOWING THE FIRST SET. THE
0023 FINAL CARD OF THE DATA DECK IS 888888886NN8888. IF MODE = 2, AS A-
0024 BOVE, THE PROGRAM IS DEALING WITH AN ACTUAL INPUT AND PRODUCING
0025 OPTIMAL OUTPUT. HOWEVER, FOR TEST AND RESEARCH PURPOSES, IT MAY BE
0026 USEFUL TO USE MODE = 1, IN WHICH DATA ARE GENERATED AND RANDOMLY
0027 MODIFIED SO AS TO PROVIDE PROBLEMS FOR SOLUTION. DETAILS OF THIS
0028 USE MAY BE FOUND IN THE THESIS REFERRED TO ABOVE.
0029
0030 MOST MATRICES ARE REGARDED AS DOUBLE SUBSCRIPT ARRAYS AS FOLLOWS:
0031 IF THERE ARE M ROWS AND N COLUMNS, THE ARRAY HAS DIMENSION (M+1,N)
0032 LET E(I,J) REFER TO THE ARRAY AND, LEFT A(I,J) REFER TO THE MATRIX,
0033 THEN E(1,1) IS AN INTEGER IDENTIFYING THE ARRAY AND THE MATRIX,
0034 E(1,2) AND E(1,3) ARE INTEGERS M AND N, RESPECTIVELY. FOR THE
0035 REMAINDER OF THE ELEMENTS, E(I+1,J) = A(I,J), THE OTHER MATRI-
0036 CES THAT HAVE DIMENSIONS (N,N) ARE USED HERE AS SCRATCH MATRICES.
0037
0038 CLOCK=ITIME(0)*.01
0039
0040 5 NUMBER=0
0041 CLUCK=ITIME(1)*.01
0042
0043 IF(MODE=1)1000,1000,1250
0044 CALL REDEM(P,1)
0045 WRITE(6,50)
0046 1000
0047 50 FORMAT(19H1 MATRIX P FOLLOWS)

```

CC


```

0048 CALL PRINM(P,6)
0049 CALL REDEM(8E,2)
0050 WRITE(6,55)
0051 55 FORMAT(20H0 MATRIX BE FOLLOWS)
0052 MM=BE(1,3)+.1
0053 LL=MM+1
0054 CALL PRINM(BE,MM)
0055 CALL ULTMA(EB,P,BE,7,MM,6,7)
0056 WRITE(6,60)
0057 60 FORMAT(20H0 MATRIX EB FOLLOWS)
0058 CALL PRINM(EB,MM)
0059 CALL BLANCA(U,BE,MM)
0060 CALL BLANCA(V,EB,MM)
0061 GO TO 1300
0062 CALL REDEM(U,1)
0063 MM=U(1,3)+.1
0064 LL=MM+1
0065 CALL REDEM(V,2)
0066 CONTINUE
0067 WRITE(6,65)
0068 65 FORMAT(19H0 MATRIX U FOLLOWS)
0069 CALL PRINM(U,MM)
0070 WRITE(6,70)
0071 70 FORMAT(19H0 MATRIX V FOLLOWS)
0072 CALL PRINM(V,MM)
0073 CALL TRANM(TR,U,LL,MM)
0074 CALL ULTMA(P,U,TR,7,6,MM,LL)
0075 CALL TRANM(TR,V,LL,MM)
0076 CALL ULTMA(Q,V,TR,7,6,MM,LL)
0077 CALL ULTMA(RCOT,U,TR,7,6,MM,LL)
0078 CALL TRANM(GG,ROOT,7,6)
0079 CALL ADDEM(Y,GG,ROOT,1.00,7,6)
0080 CALL BIGSM(W,P)
0081 NT=W(1,2)+.1
0082 NL=NT+1
0083 CALL AINVM(S,W,Z,X,NL,NT)
0084 CALL BIGCM(CCL,Y)
0085 CALL ULTMB(AA,S,COL,NL,NT)
0086 CALL SQUAM(QC,AA)
0087 CLOCK=ITIME(0)*.01-CLOCK
0088 WRITE(6,2000)CLOCK
0089 2000 FORMAT(18H0 ELAPSED TIME = ,F12.2)
0090 WRITE(6,105)
0091 105 FORMAT(42H0 OUTPUT NO. 1,MINIMIZES V-SQUARE ERROR )
0092 CALL AINVM(GG,QQ,AB,ZZ,7,6)
0093 CALL OUTPU(QC,GG,U,V,BE,EB,TR,T2,LL,MM)
0094 CALL BIGSM(W,Q)
0095

```

```

2100 CALL AINV(M(S,W,X,Z,NL,NT))
      CALL BIGCM(CCL,Y)
      CALL ULTMB(AA,S,COL,NL,NT)
      CALL SQUAM(Y,AA)
      CALL AINV(M(PI,Y,AB,ZZ,7,6)
      CLOCK=ITIME(O)*.01-CLOCK
      WRITE(6,2100)CLOCK
      FORMAT(18H0 ELAPSED TIME = ,F12.2)
      CLOCK=ITIME(C)*.01
      WRITE(6,106)
106  FORMAT(42H0 OUTPUT NO. 2, MINIMIZES U-SQUARE ERROR )
      CALL OUTPU(PI,Y,U,V,BE,EB,TR,T2,LL,MM)
      CALL ULTMA(Y,Q,P,7,6,6,7)
      CALL ULTMA(Y,Q,P,7,6,6,7)
      CALL ULTMA(BCOT,PI,P,7,6,6,7)
      CALL SQRCOT(ROOT,BOOT,Y)
      CALL AINV(M(QC,P,AB,ZZ,7,6)
      CALL ULTMA(Y,ROOT,QQ,7,6,6,7)
      CALL AINV(M(RCOT,Y,AB,ZZ,7,6)
      CLOCK=ITIME(O)*.01-CLOCK
      WRITE(6,2200)CLOCK
      FORMAT(18H0 ELAPSED TIME = ,F12.2)
      CLOCK=ITIME(O)*.01
      WRITE(6,104)
104  FORMAT(42H0 OUTPU NO. 3A, MINIMIZES TRIANGLE ERROR )
      CALL OUTPU(Y,ROOT,U,V,EB,BE,T2,TR,LL,MM)
      CALL ULTMA(GG,P,Q,7,6,6,7)
      CALL ULTMA(BCOT,P,PI,7,6,6,7)
      CALL SQRCOT(ROOT,BOOT,GG)
      CALL ULTMA(GG,QQ,ROOT,7,6,6,7)
      CALL AINV(M(ROOT,GG,AB,ZZ,7,6)
      CLOCK=ITIME(O)*.01-CLOCK
      WRITE(6,2300)CLOCK
      FORMAT(18H0 ELAPSED TIME = ,F12.2)
      CLOCK=ITIME(C)*.01
      WRITE(6,300)
300  FORMAT(42H0 OUTPUT NO. 3B, MINIMIZES TRIANGLE ERROR )
      CALL OUTPU(GG,ROOT,U,V,EB,BE,T2,TR,LL,MM)
      DO 400 I=2,7
      DO 400 J=1,6
      CONTINUE
      CALL AINV(M(RCOT,BOOT,AB,ZZ,7,6)
      CLOCK=ITIME(O)*.01-CLOCK
      WRITE(6,2400)CLOCK
      FORMAT(18H0 ELAPSED TIME = ,F12.2)
      CLOCK=ITIME(O)*.01
      WRITE(6,500)

```

0096
0097
0098
0099
0100
0101
0102
0103
0104
0105
0106
0107
0108
0109
0110
0111
0112
0113
0114
0115
0116
0117
0118
0119
0120
0121
0122
0123
0124
0125
0126
0127
0128
0129
0130
0131
0132
0133
0134
0135
0136
0137
0138
0139
0140
0141
0142
0143

```

500  FORMAT(42H0  OUTPUT NO.3C, MINIMIZES TRIANGLE ERROR )
      CALL OUTPU(BCOT,ROOT,U,V,EB,RE,T2,TR,LL,MM)
      CALL SQRRT(Y,P,Q)
      CALL AINV(M(PI,Y,AB,ZZ,7,6))
      WRITE(6,200)
200  FORMAT(42H0  OUTPUT NO.3D, MINIMIZES TRIANGLE ERROR )
      CALL OUTPU(Y,PI,U,V,EB,RE,T2,TR,LL,MM)
      CLOCK=ITIME(C)*.01-CLOCK
      WRITE(6,250)CLOCK
250  FORMAT(18H0  ELAPSED TIME = ,F12.2)
      CLOCK=ITIME(C)*.01
      CLUCK=ITIME(I)*.01-CLOCK
      WRITE(6,300)CLUCK
300  FORMAT(24H0  TOTAL ELAPSED TIME = ,F12.2)
      GO TO 5
      END
0144
0145
0146
0147
0148
0149
0150
0151
0152
0153
0154
0155
0156
0157
0158
0159

```


A-3 Listing of Program CECI 2.

The cards of program CECI 2 are numbered from 00000 to 01570.

```

00000
00010
00020
00030
00040
00050
00060
00070
00080
00090
00100
00110
00120
00130
00140
00150
00160
00170
00180
00190
00200
00210
00220
00230
00240
00250
00260
00270
00280
00290
00300
00310
00320
00330
00340
00350
00360
00370
00380
00390
00400
00410
00420
00430
00440
00450
00460
00470

PROGRAM CECI 2 "SOLVES" THE OVERDETERMINED LINEAR SYSTEM XU=V, GIVEN
WHERE X IS AN UNKNOWN SYMMETRIC (6,6) MATRIX AND U AND V ARE THE
RECTANGULAR MATRICES. ONE SOLUTION IS GIVEN, WHICH SOLVES THE
PROBLEM IN THE SENSE OF SATISFYING AN ERROR CRITERION. FOR DE-
TAILS SEE NAVAL POSTGRADUATE SCHOOL THESIS, "DETERMINATION OF
OPTIMAL COMPLIANCE AND STIFFNESS MATRICES FROM EXPERIMENTAL DATA"
BY LT. C. F. VILLARAN, 1967. CARD NUMBER READS INPUT MATRICES U
1 OR MODE = 2. IF MODE = 2, THE PROGRAM READS INPUT MATRICES U
AND V FROM PUNCHED CARDS ACCORDING TO FORMAT (5F14.6). READING IS
BY COMPLETE ROW, A NEW ROW MATRIX U. THE FIRST CARD FOR EACH MA-
TRIX V FOLLOWS: 3BFB8B8BNN3B8B WHERE B DENOTES BLANK, NN IS THE
IS AS FOLLOWS: 3BFB8B8BNN3B8B WHERE B DENOTES BLANK, NN IS THE
TWO DIGIT NUMBER OF COLUMNS IN U AND V, NN NOT GREATER THAN 50,
AND M = 1 FOR MATRIX U, M = 2 FOR MATRIX V. CARDS FOR MATRICES U
AND V IMMEDIATELY FOLLOW THE PROGRAM. IF THERE IS MORE THAN ONE SET OF
//GO SYSDD*) IN THE PROGRAM. IF THERE IS MORE THAN ONE SET OF
INPUTS, THESE ARE PLACED IN ORDER FOLLOWING THE FIRST SET. THE
FINAL CARD OF THE DATA DECK IS B3B8B8B8B8B8B6. IF MODE = 2, AS A-
BOVE, THE PROGRAM IS DEALING WITH AN ACTUAL INPUT AND PRODUCING
OPTIMAL OUTPUT. HOWEVER, FOR TEST AND RESEARCH PURPOSES, IT MAY BE
USEFUL TO USE MODE = 1, IN WHICH DATA ARE GENERATED AND RANDOMLY
MODIFIED SO AS TO PROVIDE PROBLEMS FOR SOLUTION. DETAILS OF THIS
USE MAY BE FOUND IN THE THESIS REFERRED TO ABOVE.

MCST MATRICES ARE REGARDED AS DOUBLE SUBSCRIPT ARRAYS AS FOLLOWS;
IF THERE ARE M ROWS AND N COLUMNS, THE ARRAY HAS DIMENSION (M+1,N)
LET E(I,J) REFER TO THE ARRAY AND LET A(I,J) REFER TO THE MATRIX,
THEN E(1,1) IS AN INTEGER IDENTIFYING THE ARRAY AND THE MATRIX,
E(1,2) AND E(1,3) ARE INTEGERS M AND N, RESPECTIVELY. FOR THE
REMAINDER OF THE ELEMENTS, E(I+1,J) = A(I,J), THE OTHER MATRI-
CES THAT HAVE DIMENSIONS (N,N) ARE USED HERE AS SCRATCH MATRICES.

CARD NUMBER C0870 IS AN SCALAR QUANTITY CALLED BETA THAT DESIGNA-
TE THE RELATIVE IMPORTANCE OF THE TWO SEPARATE CRITERIA OF WHICH
ALFA 4 IS COMPOSED.

IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
DIMENSION T2(51,6), Z(21,21), X(21,21), AR(6,6), Z(6,6), BE(7,50),
1EB(7,50), U(7,50), V(7,50), TR(51,6), P(7,6), Y(7,5), ROOT(7,6), Q(7,6),
2W(22,21), S(22,21), COL(22), AA(22), QQ(7,6), GG(7,6), PP(7,6), R(7,6),
3RR(7,6), T(7,6), TT(7,6), PI(7,6), SS(22,21)
COMMON NUMBER
CLCCK=ITIME(C)*.C1
CLUCK=0
CLUCK=ITIME(1)*.C1
MODE=1

```

5

CC

```

1000 IF(MODE-1)IGCC,1000,1250
      CALL REDEM(P,1)
      WRITE(6,50)
50   FORMAT(19H1 MATRIX P FOLLOWS)
      CALL PRINM(P,6)
      CALL REDEM(BE,2)
      WRITE(6,55)
55   FORMAT(20H0 MATRIX RE FOLLOWS)
      MM=BE(1,3)+.1
      LI=MM+1
      CALL PRINM(BE,MM)
      CALL ULTMA(EB,P,BE,7,MM,6,7)
      WRITE(6,60)
60   FORMAT(20H0 MATRIX EB FOLLOWS)
      CALL PRINM(EB,MM)
      CALL BLANCA(L,BE,MM)
      CALL BLANCA(V,EB,MM)
      GO TO 1300
1250 CALL REDEM(U,1)
      MM=U(1,3)+.1
      LL=MM+1
      CALL REDEM(V,2)
1300 CCNTINUE
      WRITE(6,65)
65   FORMAT(19H0 MATRIX U FOLLOWS)
      CALL PRINM(U,MM)
      WRITE(6,70)
70   FORMAT(19H0 MATRIX V FOLLOWS)
      CALL PRINM(V,MM)
      CALL TRANM(TR,U,LL,MM)
      CALL ULTMA(P,U,TR,7,6,MM,LL)
      CALL TRANM(TR,V,LL,MM)
      CALL ULTMA(G,V,TR,7,6,MM,LL)
      CALL ULTMA(RCOT,U,TR,7,6,MM,LL)
      CALL TRANM(GG,ROCT,7,6)
      CALL ADDEM(Y,GG,ROCT,1,D0,7,6)
      CALL BIGSM(W,P)
      NT=W(1,2)+.1
      NL=NT+1
      BETA=C.800
      CALL AINVW(S,W,Z,X,NL,NT)
      CALL BIGCM(CCL,Y)
      CALL ULTMB(AA,S,COL,NL,NT)
      CALL SQUAM(QC,AA)
      CALL BIGSM(W,Q)
      CALL AINVW(SS,A,X,Z,NL,NT)
      CALL ULTMB(AA,SS,COL,NL,NT)
      CALL SQUAM(GC,AA)
00480
00490
00500
00510
00520
00530
00540
00550
00560
00570
00580
00590
00600
00610
00620
00630
00640
00650
00660
00670
00680
00690
00700
00710
00720
00730
00740
00750
00760
00770
00780
00790
00800
00810
00820
00830
00840
00850
00860
00870
00880
00890
00900
00910
00920
00930
00940
00950

```

```

      CALL AINVMM(PI,GG,AR,ZZ,7,6)
      TETA=1.D-1
      DO 80 I=2,7
      DO 80 J=1,6
      QQ(I,J)=(QQ(I,J)+TETA*PI(I,J))/(1.D0+TETA)
      CONTINUE
      CLOCK=ITIME(C)*.01-CLOCK
      WRITE(6,2000)CLOCK
      FORMAT(18H0 ELAPSED TIME = ,F12.2)
      CLOCK=ITIME(C)*.01
      RCL=0.D0
      DO 201 DO 235 K=1,100
      TOL=K
      CALL AINVMM(GG,QQ,AR,ZZ,7,6)
      CALL ULTMA(PP,Q,GG,7,6,6,7)
      C
      CALL ULTMA(R,GG,Q,7,6,6,7)
      CALL ADDEM(RR,PP,R,1.D0,7,6)
      CALL ADDEM(PP,RR,Y,-1.D0,7,6)
      CALL ULTMA(R,GG,PP,7,6,6,7)
      CALL ULTMA(RR,R,GG,7,6,6,7)
      CALL ADDEM(TT,Y,RR,TETA,7,6)
      CALL BIGCM(CCL,TT)
      CALL ULTMB(AA,S,COL,NL,NT)
      CALL SQUAM(PP,AA)
      CALL ADDEM(R,QQ,PP,-1.D0,7,6)
      C
      DC 210 I=2,7
      DC 210 J=1,6
      RET=DMAX1(DABS(R(I,J)))
      CONTINUE
      DO 220 I=2,7
      DO 220 J=1,6
      QQ(I,J)=PP(I,J)
      CONTINUE
      IF(RET-1.D-5)240,240,235
      235 CCNTINUE
      C
      RCL=RCL+TCL
      TETA=TETA+1.D-1
      IF(RET-TETA)202,203,203
      GO TO 201
      203 CONTINUE
      CALL AINVMM(GG,QQ,AR,ZZ,7,6)
      CLOCK=ITIME(C)*.01-CLOCK
      WRITE(6,2100)CLOCK
      FORMAT(18H0 ELAPSED TIME = ,F12.2)
      CLOCK=ITIME(C)*.01
      2100

```

```

00960
00970
00980
00990
01000
01010
01020
01030
01040
01050
01060
01070
01080
01090
01100
01110
01120
01130
01140
01150
01160
01170
01180
01190
01200
01210
01220
01230
01240
01250
01260
01270
01280
01290
01300
01310
01320
01330
01340
01350
01360
01370
01380
01390
01400
01410
01420
01430

```

```

238 WRITE(6,238)ROL
    FORMAT(26HC NUMBER OF INTERACTIONS = , D14.8)
105 WRITE(6,105)
    FORMAT(53HC OUTPUT NO.4, MINIMIZES V + BETA * U SQUARE ERRORS )
    CALL OUTPUT(QQ,GG,U,V,BE,ER,TR,T2,LL,MM,BETA)
    CLOCK=ITIME(0)*.C1-CLOCK
2200 WRITE(6,2200)CLOCK
    FORMAT(18HC ELAPSED TIME = ,F12.2)
    CLOCK=ITIME(0)*.01
    CLUCK=ITIME(1)*.C1-CLUCK
3000 WRITE(6,3000)CLUCK
    FORMAT(24HC TOTAL ELAPSED TIME = ,F12.2)
    GO TO 5
END

```

```

01440
01450
01460
01470
01480
01490
01500
01510
01520
01530
01540
01550
01560
01570

```

APPENDIX B

SUBROUTINE PROGRAMS

B-1 Description of Subroutines.

SUBROUTINE ADDEM (A,B,C,D,M,L) provides the result of $A = B + DC$, where D is a scalar constant, and A, B, and C are double precision matrices dimensioned (M,L), where M and L are not less than 3. With $D = 1.0$, $A = B + C$; with $D = -1.0$, $A = B - C$.

The matrices B and C from the calling program must have identical order, otherwise a message is printed and the execution is halted.

SUBROUTINE AINVM (A,B,C,D,L,N) forms the inverse $A = B^{-1}$ of input matrix B. Matrices A, B, C, and D are double precision, with A and B dimensioned (L,N). Matrices C and D, which are used as scratch matrices, have dimension (N,N), where $L = N + 1$, and N is not less than 3.

A message is printed if the input matrix B is non-square and the execution is halted.

SUBROUTINE BIGCM (COL,Q) forms the 21-element vector COL from the 6x6 matrix Q, according to the scheme described in Section 3; if the input matrix Q is non-square a message is printed and the execution is halted.

SUBROUTINE BIGSM (BIG,P) forms the 21 x 21 matrix BIG, from the 6x6 matrix P according to the scheme described in Section 3; if the input matrix P is non-square a message is printed and the execution is halted.

SUBROUTINE BLANCA (CO,BE,N) forms matrix CO from matrix BE by introducing random errors in the latter according to the method described in Section 7. The arrays CO and BE are dimensioned (7,N). The maximum relative error, denoted by the symbol r in the exposition in Section 7, is governed by the computer variable PERC, which is one tenth the relative error. If $PERC = 1.D-2 = .0100\dots$, then $r = .100\dots$, and a maximum of

10% error is introduced. PERC may be changed by changing card number 0313 in the subroutine.

SUBROUTINE JACVAT (B,N,NOYES, RR,SS,EIRU,NDIM) gives the eigenvalues and eigenvectors of the symmetric matrix B. If B is non-symmetrical the subroutine first makes B symmetric by forming $\frac{1}{2}B + \frac{1}{2}B^T$. RR is the spectral matrix that contains the eigenvalues of B. SS is the modal matrix of B that contains the eigenvectors. EIRU is the square root of the elements in the diagonal matrix RR. N is the dimension of B. NOYES is 0 or 1; if 0, no eigenvalues will be recovered; if 1, the eigenvalues will be recovered in matrix RR. In the application made in this thesis, NDIM is the same as N. All the matrices and variables are in double precision.

SUBROUTINE MAUSS3 (N,EP,A,X,KER) is a double precision FORTRAN IV language subroutine which inverts a symmetric matrix. A is the matrix to be inverted, X is its inverse. Both have dimension of (N,N). EP is a tolerable error, taken as 1.D-20 in the present work. KER is an output flag having the value 1, if all has gone well, or 2, if there has been any difficulty of execution.

SUBROUTINE OUTPU (R,B,U,V,T1,T2,T3,T5,LL,L) has inputs R, B, U, V, where R is the desired compliance matrix. B is the desired stiffness matrix, and U, V are the data for the problem (such that $RU = V$, approximately). It prints the matrix R, and computes and prints the three criterion numbers α_i ($i = 1,2,3$). Matrices T1-T5 that appear in the calling statement are scratch matrices. LL and L are dummy variables used in the dimension of the matrices and are related to the number n of vector pairs used.

SUBROUTINE PRINM (A,L) causes the (6,L) matrix A to be printed, where L is not less than 3. The first line printed contains the following information: MATRIX NUMBER XX, YY ROWS, ZZ COLUMNS. This is

followed by a print of the matrix, line by line, in FORMAT (1X, 8D12.4).

SUBROUTINE RECIP (A,B) A and B are diagonal matrices (6,6) and $A = B^{-1}$, where B is the input matrix.

SUBROUTINE REDEM (A,IDENT) reads into the computer memory from the data deck which is punched as described in Appendix A. A is the "name" of the matrix and IDENT is an identifying integer to assure that the card-group which is read is indeed the one desired.

SUBROUTINE SQRERT (A,B,C) solves the matrix equation that satisfies criterion 3, using the procedure that twice takes the square root of a symmetric matrix. $B = UU^T$, $C = VV^T$ are the inputs, and A is the desired compliance matrix.

SUBROUTINE SQROOT (A,CC,G) takes the square root of a non-symmetric matrix by iteration procedure. Matrices CC and G are inputs. CC is the approximate square root; G is the matrix whose square root is desired, and A is the desired square root.

SUBROUTINE SQUAM (A,B) forms a 6 by 6 matrix A from the 21-element column vector B, arranging the results properly as implied in Section 3.

SUBROUTINE TRACM (T,A) forms the trace T of the 6 by 6 matrix A.

SUBROUTINE TRANM (A,B,LL,L) forms the transpose of matrix B(7,L), obtaining matrix A(LL,6), where L is no less than 3, and $LL = L + 1$.

SUBROUTINE ULTMA (A,B,C,MM,M,L,LL) forms the matrix product $A = BC$. $A(MM,M) = B(MM,L) \times C(LL,M)$, where MM, M, L, LL, can be any numbers not less than 3 for which the matrices are conformable for multiplication. A message is printed if the matrices B and C are not conformable and execution is halted.

SUBROUTINE ULTMB (A,B,C,LL,L) multiplies the square matrix B by a column matrix C, to obtain the column matrix A. $A(LL) = B(LL,L) \times C(LL)$,

where L is any number not less than 3, and $LL = L + 1$.

SUBROUTINE OUTPUT (R,B,U,V,T1,T2,T3,T5,LL,L,BETA) is the same as SUBROUTINE OUTPU. The only differences are the inclusion of parameter BETA in the calling statement, and the printing of four criterion numbers instead of three, α_i ($i = 1,2,3,4$).

B-2 Subroutine Listing

SUBROUTINE	CARDS NUMBERED	
	FROM	TO
ADDEM	0160	0182
AINVM	0183	0208
BIGCM	0209	0233
BIGSM	0234	0288
BLANCA	0289	0336
JACVAT	0337	0481
MAUSS3	0482	0532
OUTPU	0533	0559
PRINM	0560	0576
RECIP	0577	0594
REDEM	0595	0616
SQRERT	0617	0638
SQROOT	0639	0692
SQUAM	0693	0727
TRACM	0728	0744
TRANM	0745	0763
ULTMA	0764	0791
ULTMB	0792	0814
OUTPUT	20000	20360

```

C  SUBROUTINE ADDEM ADD MXL MATRICES, WHERE M=7,50 AND L=6,50
SUBROUTINE ACDEM (A,B,C,D,M,L)
REAL*8 A,B,C,D
DIMENSION A(M,L),B(M,L),C(M,L)
COMMON NUMBER+1
NUMBER=NUMBER+1
A(1,1)=NUMBER
1003 IF(DABS(B(1,2))-C(1,2))-.100)1003,1002,1002
1004 IF(DABS(B(1,3))-C(1,3))-.100)1004,1002,1002
A(1,2)=B(1,2)
A(1,3)=B(1,3)
M1=A(1,2)+1.100
N=A(1,3)+.100
DO 1005 J=2,M1
DO 1005 K=1,N
A(J,K)=B(J,K)+D*C(J,K)
1005 CONTINUE
GO TO 3334
1002 WRITE(6,1006) NUMBER
1006 FORMAT(48H NONCCNFOMABLE MATRIX ADDITION TO FORM NUMBER, I7)
STOP
3334 RETURN
END

```

```

C  SUBROUTINE AINVM INVERTS N X N MATRIX
SUBROUTINE AINVM(A,B,C,D,L,N)
IMPLICIT REAL*8(A-H,O-Z),INTEGER(I-N)
DIMENSION A(L,N),B(L,N),C(N,N),D(N,N)
COMMON NUMBER
NUMBER=NUMBER+1
1023 IF(DABS(B(1,2))-B(1,3))-1.0-D-1)1025,1023,1023
1024 WRITE(6,1024) NUMBER
3333 FORMAT(37H NONSQUARE INVERSION TO FORM NUMBER , I7)
1025 STOP
A(1,1)=NUMBER
A(1,2)=B(1,2)
A(1,3)=B(1,3)
DO 1026 J=1,N
DO 1026 K=1,N
M=J+1
D(J,K)=B(M,K)
CONTINUE
1026 CALL MAUSS3(N,1.0-20.0,D,C,K2)
DO 1027 J=2,L
DO 1027 K=1,N
JJ=J-1
A(J,K)=C(JJ,K)
CONTINUE
1027 RETURN
END

```

C	SUBROUTINE BIGCM FORMS BIG COLUMN MATRIX,COL, FROM SQUARE MATRIX,Q.	0209
	SUBROUTINE BIGCM(COL,Q)	0210
	REAL*8 COL,Q,DELTA	0211
	DIMENSION COL(22),Q(7,6)	0212
	COMMON NUMBER	0213
	NUMBER=NUMBER+1	0214
	IF(DABS(Q(1,2))-Q(1,3))-1.0-1)722,720,720	0215
720	WRITE(6,721)NUMBER	0216
721	FORMAT(39H NONSQUARE INPUT TO FORM BIGCM NUMBER , I5)	0217
	STOP	0218
722	N=Q(1,2)+.1	0219
	JJ=1	0220
	DO 723 K=1,N	0221
	DO 723 J=1,K	0222
	JJ=JJJ+1	0223
	JJ=J+1	0224
	IF(J-K)724,725,724	0225
724	DELTA=1.00	0226
	GO TO 726	0227
725	DELTA=2.00	0228
726	COL(JJJ)=Q(JJ,K)/DELTA	0229
723	CONTINUE	0230
	COL(1)=JJJ-1	0231
	RETURN	0232
	END	0233

```

C
C
SUBROUTINE BIGSM FORMS BIG SQUARE MATRIX, BIG, FROM SQUARE MATRIX
P.
SUBROUTINE BIGSM(BIG,P)
REAL*8 BIG,P,I
DIMENSION BIG(22,21),P(7,6),NA(21,21)
COMMON NUMBER
NUMBER=NUMBER+1
IF(DABS(P(1,2)-P(1,3))-1.0-1)822,820,820
820 WRITE(6,821) NUMBER
821 FORMAT(41H KONSQUARE INPUT TO FORM BIGSUM A NUMBER, I5)
STOP
822 N=P(1,2)+.1
MMM=N
NA(1,1)=1
NA(1,2)=2
DO 50 J=3,MMM
JM=J-1
JMM=J-2
NA(1,J)=1+2*NA(1,JM)-NA(1,JMM)
CONTINUE
50 DO 70 J=2,MMM
DO 60 K=2,J
KM=K-1
NA(K,J)=NA(KM,J)+1
NA(J,K)=NA(K,J)
CONTINUE
60 CONTINUE
70 DO 71 J=2,MMM
71 NA(J,1)=NA(1,J)
NT=N*(N+1)
T=NT
NP1=N+1
NNN=0.500*T+1.0-3
NNN1=NNN+1
DO 824 I=1,NNN1
DO 1824 J=1,NNN
BIG(I,J)=0.0C
CONTINUE
1824 CONTINUE
824 BIG(1,1)=NUMBER
BIG(1,2)=.5DC*T
BIG(1,3)=BIG(1,2)
NP=N+1
DO 823 I=2,NP
IM=I-1
DO 1823 J=1,N
DO 2823 K=1,N
IP=NA(K,IM)+1

```

0234
0235
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247
0248
0249
0250
0251
0252
0253
0254
0255
0256
0257
0258
0259
0260
0261
0262
0263
0264
0265
0266
0267
0268
0269
0270
0271
0272
0273
0274
0275
0276
0277
0278
0279
0280
0281

```

      JP=NA(K,J)
      BIG(IP,JP)=BIG(IP,JP)+P(I,J)
2823  CONTINUE
1823  CONTINUE
      823  CONTINUE
      RETURN
      END

```

```

0282
0283
0284
0285
0286
0287
0288

```



```

C SUBROUTINE BLANCA GIVE RANDOM VALUES FOR MATRIX, CO=OUTPUT, BE=INPUT
SUBROUTINE BLANCA(CO,BE,N)
IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
DIMENSION CO(7,N), BE(7,N)
COMMON NUMBER
NUMBER=NUMBER+1
CO(1,1)=NUMBER
CO(1,2)=BE(1,2)
CO(1,3)=BE(1,3)
NN=BE(1,3)
RHO=NUMBER
DO 5 I=2,7
P=I
DO 5 J=1,NN
Q=J
RHO=RHO+3.632700
IF(RHO-9.99900)12,11,11
11 RHO=RHO-8.415300
12 XX=RHO*(P/1.001+Q/2.501+3.141600/5.00)
KX=XX
AKX=KX
BKX=XX-AKX
CKX=1.01*8KX
TT=BE(I,J)
PERC=0.00
QQ=PERC*CKX*TT
KKX=CKX
DKKX=KKX
EKKX=DKKX/2.00
LKKX=EKKX
GKKX=LKKX
OKKX=EKKX-GKKX
IF(TT)15,10,15
10 RR=1.01*PERC*CKX
6 IF(OKKX-5.0-D-1)6,7,6
6 CO(I,J)=RR
7 GO TO 5
7 CO(I,J)=--RR
7 GO TO 5
15 IF(OKKX-5.0-D-1)3,1,3
3 CO(I,J)=TT+QQ
1 CO(I,J)=TT-QQ
5 CONTINUE
222 WRITE(6,222) PERC
FORMAT(20H0 ERROR IN BLANCA = ,F10.5 )
RETURN
END

```

```

C      IN JACVAT B(7,6)=A(6,6) IS THE SQUARE MATRIX, FOR OUR PORPOUSE A IS
C      SYMETRIX AND POSITIVE DEFINITE, N IS THE DIMENSION OF A, MOVES=1,
C      RR(7,6)=EIVU(6,6) IS THE MATRIX OF EIGENVALUES(RR=SPECTRAL), SS(7,6)=
C      EIVR(6,6) IS THE MODAL MATRIX(EIGENVECTORS), EIRU(7,6) IS THE SQUARE
C      ROOT OF THE ELEMENTS IN DIAGONAL EIVU, NDIM IS THE ACTUAL NUMERICAL
C      VALUE OF N.
C      SUBROUTINE JACVAT(B,N,MOVES,RR,SS,EIRU,NDIM)
C      DIMENSION B(7,6),EIVU(6),RR(7,6),EIVR(6,6),EIRU(7,6),A(7,6),
C      1SS(7,6)
C      1U,RR,B,SS
C      COMMON NUMBER
C      DO 200 I=1,N
C      DO 200 J=1,N
C      A(I,J)=B(I+1,J)
C      200 CONTINUE
C      99 IF(NOVES)99,102,99
C      CONTINUE
C      DO 101 J=1,N
C      DO 100 I=1,N
C      EIVR(I,J)=0.00
C      101 EIVR(J,J)=1.00
C      102 ATOP=0.00
C      DO 112 J=1,N
C      DC 111 I=1,J
C      IF(A(I,J))-A(J,I))90,103,90
C      90 A(I,J)=0.500*(A(I,J)+A(J,I))
C      A(J,I)=A(I,J)
C      103 CONTINUE
C      IF(ATOP-DABS(A(I,J)))104,111,111
C      104 ATOP=DABS(A(I,J))
C      111 CONTINUE
C      112 EIVU(J)=A(J,J)
C      IF(ATOP)109,109,113
C      109 WRITE(6,110)
C      110 FORMAT(26H IN JACVAT, MATRIX A = 0 )
C      RETURN
C      113 AVGF=DFLOAT(N+(N-1))*5.50-1
C      D=0.00
C      DO 114 JJ=2,N
C      DO 114 II=2,JJ
C      S=A(II-1,JJ)/ATOP
C      S=S*S+D
C      114 DSTOP=(1.0-14)*D
C      THRESH=DSQRT(C/AVGF)*ATOP
C      115 IFLAG=0
C      DO 130 JCOL=2,N
C      JCOL1=JCOL-1

```

```

0337
0338
0339
0340
0341
0342
0343
0344
0345
0346
0347
0348
0349
0350
0351
0352
0353
0354
0355
0356
0357
0358
0359
0360
0361
0362
0363
0364
0365
0366
0367
0368
0369
0370
0371
0372
0373
0374
0375
0376
0377
0378
0379
0380
0381
0382
0383
0384

```

```

DO 130 IROW=1,JCOL1
AIJ=A(IROW,JCOL)
IF(DABS(AIJ)-THRESH)130,130,117
117 AII=A(IROW,IROW)
AJJ=A(JCOL,JCOL)
S=AJJ-AII
IF(DABS(AIJ)-1.0-18*DABS(S))130,130,118
118 IFLAG=1
119 IF(1.0-20*DABS(AIJ)-DABS(S))116,119,119
S=5.0-1
S=DSQRT(S)
C=S
GO TO 120
116 T=AIJ/S
S=0.2500/DSQRT(0.2500+T*T)
C=DSQRT(0.500+S)
S=2.00*T*S/C
120 DO 121 I=1,IROW
T=A(I,IROW)
U=A(I,JCOL)
A(I,IROW)=C*T-S*U
121 A(I,JCOL)=S*T+C*U
I2=IROW+2
IF(I2-JCOL)127,127,123
127 CONTINUE
DO 122 I=12,JCOL
T=A(I-1,JCOL)
U=A(IROW,I-1)
A(I-1,JCOL)=S*U+C*T
122 A(IROW,I-1)=C*U-S*T
123 A(JCOL,JCOL)=S*AIJ+C*AJJ
A(IROW,IROW)=C*A(IROW,IROW)-S*(C*AIJ-S*AJJ)
DO 124 J=JCOL,N
T=A(IROW,J)
U=A(JCOL,J)
A(IROW,J)=C*T-S*U
124 A(JCOL,J)=S*T+C*U
131 IF(NOVES)131,126,131
DO 125 I=1,N
T=EIVR(I,IROW)
EIVR(I,IROW)=C*T-EIVR(I,JCOL)*S
125 EIVR(I,JCOL)=S*T+EIVR(I,JCOL)*C
126 CONTINUE
S=AIJ/ATOP
D=0-S*S
IF(D-DSTOP)1260,129,129
1260 D=0.00

```

0433
0434
0435
0436
0437
0438
0439
0440
0441
0442
0443
0444
0445
0446
0447
0448
0449
0450
0451
0452
0453
0454
0455
0456
0457
0458
0459
0460
0461
0462
0463
0464
0465
0466
0467
0468
0469
0470
0471
0472
0473
0474
0475
0476
0477
0478
0479
0480
0481

```

DO 128 JJ=2,N
DO 128 II=2,JJ
S=A(II-1,JJ)/ATOP
128 D=S*S+D
129 DSTOP=(1.D-14)*D
130 THRESH=DSQRT(C/AVGF)*ATOP
CONTINUE
IF(IFLAG)115,134,115
134 T=A(1,1)
EIVU(1)=T
DO 132 J=2,N
T=A(J,J)
EIVU(J)=T
DO 132 I=2,J
A(I-1,J)=A(J,I-1)
132 DO 25 I=1,N
DO 25 J=1,N
SS(I+1,J)=EIVR(I,J)
25 CONTINUE
NUMBER=NUMBER+1
SS(1,1)=NUMBER
SS(1,2)=N
SS(1,3)=N
DO 1300 I=1,7
DO 1300 J=1,6
RR(I,J)=0.DO
1300 CONTINUE
DO 1350 I=1,6
RR(I+1,I)=EIVU(I)
1350 CONTINUE
NUMBER=NUMBER+1
RR(1,1)=NUMBER
RR(1,2)=N
RR(1,3)=N
DO 1400 I=1,7
DO 1400 J=1,6
EIRU(I,J)=C.DO
1400 CONTINUE
DO 1500 I=1,N
EIRU(I+1,I)=DSQRT(RR(I+1,I))
1500 CONTINUE
NUMBER=NUMBER+1
EIRU(1,1)=NUMBER
EIRU(1,2)=N
EIRU(1,3)=N
133 RETURN
END

```

```

SUBROUTINE MAUSS3(N,EP,A,X,KER)
REAL*8 A,X,Z,RATIO,S
DIMENSION A(N,N),X(N,N)
DO 1 I=1,N
DO 1 J=1,N
1 X(I,J)=0.00
DO 2 K=1,N
2 X(K,K)=1.00
10 DO 34 L=1,N
KP=0
Z=0.00
DO 12 K=L,N
IF(Z-DABS(A(K,L)))11,12,12
11 Z=DABS(A(K,L))
KP=K
12 CONTINUE
IF(L-KP)13,20,20
13 DO 14 J=L,N
Z=A(L,J)
A(L,J)=A(KP,J)
14 A(KP,J)=Z
DO 15 J=1,N
Z=X(L,J)
X(L,J)=X(KP,J)
15 X(KP,J)=Z
20 IF(DABS(A(L,L))-EP)50,50,30
30 IF(L-N)31,34,34
31 LPI=L+1
DO 36 K=LPI,N
IF(A(K,L))32,36,32
32 RATIO=A(K,L)/A(L,L)
DO 33 J=LPI,N
A(K,J)=A(K,J)-RATIO*A(L,J)
33 A(K,J)=A(K,J)-RATIO*X(L,J)
35 X(K,J)=X(K,J)-RATIO*X(L,J)
36 CONTINUE
34 DO 43 I=1,N
II=N+1-I
DO 43 J=1,N
S=0.00
IF(II-N)41,43,43
41 IPI=II+1
DO 42 K=IPI,N
S=S+A(II,K)*X(K,J)
42 X(II,J)={X(II,J)-S}/A(II,II)
43 KER=1
RETURN
50 KER=2
RETURN
END

```

0482
0483
0484
0485
0486
0487
0488
0489
0490
0491
0492
0493
0494
0495
0496
0497
0498
0499
0500
0501
0502
0503
0504
0505
0506
0507
0508
0509
0510
0511
0512
0513
0514
0515
0516
0517
0518
0519
0520
0521
0522
0523
0524
0525
0526
0527
0528
0529
0530
0531
0532

```

SUBROUTINE OUTPU(R,8,U,V,T1,T2,T3,T5,LL,L)
IMPLICIT REAL*8(A-H,O-Z),INTEGER(I-N)
DIMENSION R(7,6),B(7,6),U(7,L),V(7,L),T1(7,L),T2(7,L),T3(LL,6),
1T4(7,6),O(5),T5(LL,6)
WRITE(6,32)
32 FORMAT(19H0 DESIRED MATRIX IS )
CALL PRINM(R,6)
CALL ULTMA(T1,R,U,7,L,6,7)
CALL ADDEM(T2,T1,V,-1,D0,7,L)
CALL TRANM(T3,T2,LL,L)
CALL ULTMA(T4,T2,T3,7,6,L,LL)
CALL TRACM(A,T4)
O(1)=A
CALL ULTMA(T2,8,V,7,L,6,7)
CALL ADDEM(T1,U,T2,-1,D0,7,L)
CALL TRANM(T5,T1,LL,L)
CALL ULTMA(T4,T1,T5,7,6,L,LL)
CALL TRACM(A,T4)
O(2)=A
CALL ULTMA(T4,T1,T3,7,6,L,LL)
CALL TRACM(A,T4)
O(3)=A
DO 30 I=1,3
30 WRITE(6,31)I,O(I)
31 FORMAT(20H0 CRITERION NUMBER ,I4,8H EQUALS ,1PE14.6)
RETURN
END

C SUBROUTINE PRINM, PRINT 6XN MATRIX, WHERE N=6,50
SUBROUTINE PRINM(A,L)
REAL*8 A
DIMENSION A(7,L)
NT1=A(1,1)+.1
NT2=A(1,2)+.1
NT3=A(1,3)+.1
WRITE(6,1020)NT1,NT2,NT3
M1=A(1,2)+1.1
N=A(1,3)+.1
DO 1021 J=2,M1
WRITE(6,1022)(A(J,K),K=1,N)
1021 CONTINUE
1022 FORMAT(1X,8D12.4)
1020 MATRIX NUMBER , I7,3H, ,I2,6H ROWS,,I2,8H COLUMNS/)
RETURN
END

```


C SUBROUTINE RECIP INVERSE A DIAGONAL MATRIX (6X6)

0577
0578
0579
0580
0581
0582
0583
0584
0585
0586
0587
0588
0589
0590
0591
0592
0593
0594

```

SUBROUTINE RECIP(A,B)
  REAL*8 A,B
  DIMENSION A(7,6),B(7,6)
  COMMON NUMBER
  NUMBER=NUMBER+1
  A(1,1)=NUMBER
  A(1,2)=B(1,2)
  A(1,3)=B(1,3)
  DO 5 I=2,7
  DO 5 J=1,6
  IF(I-I-J)10,15,1C
  10 GO TO 5
  15 A(I,J)=1.00/B(I,J)
  5 CONTINUE
  RETURN
  END

```

C SUBROUTINE REDEM LOADS A MATRIX FROM CARDS

0595
0596
0597
0598
0599
0600
0601
0602
0603
0604
0605
0606
0607
0608
0609
0610
0611
0612
0613
0614
0615
0616

```

SUBROUTINE REDEM(A,IDENT)
  REAL*8 A
  DIMENSION A(7,50)
  COMMON NUMBER
  1040 FORMAT(6X,212,14)
  1041 READ(5,1040)M,N,JIDENT
  1043 IF(IDENT-JIDENT)1C43,1044,1043
  1045 WRITE(6,1045)NUMBER,IDENT,JIDENT
  110H READ MIXUP TO FORM NUMBER ,17,10H IDENT = ,14,
  JDEN = ,14)
  3333 STOP
  1044 M1=M+1
  1042 DO 1042 J=2,M1
  READ(5,1041) (A(J,K),K=1,N)
  NUMBER=NUMBER+1
  A(1,1)=NUMBER
  A(1,2)=M
  A(1,3)=N
  RETURN
  END

```

```

SUBROUTINE SQRRT(A,B,C)
IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
DIMENSION A(7,6),B(7,6),C(7,6),SS(7,6),P(7,6),S(7,6),E(7,6),
1EE(7,6),D(7,6),CC(7,6),Q(7,6),G(7,6),F(7,6),T(7,6),RR(7,6)
COMMON NUMBER
CALL JACVAT(B,6,1,SS,P,S,6)
CALL ULTMA(E,P,S,7,6,6,7)
CALL TRANM(EE,E,7,6)
CALL ULTMA(SS,C,E,7,6,6,7)
CALL ULTMA(D,EE,SS,7,6,6,7)
CALL JACVAT(C,6,1,SS,CC,Q,6)
CALL ULTMA(SS,CC,Q,7,6,6,7)
CALL TRANM(G,CC,7,6)
CALL ULTMA(F,SS,G,7,6,6,7)
CALL RECIP(T,S)
CALL TRANM(T,P,7,6)
CALL ULTMA(S,P,T,7,6,6,7)
CALL ULTMA(RR,S,F,7,6,6,7)
CALL ULTMA(P,RR,T,7,6,6,7)
CALL ULTMA(A,P,T,7,6,6,7)
RETURN
END

```

```

0617
0618
0619
0620
0621
0622
0623
0624
0625
0626
0627
0628
0629
0630
0631
0632
0633
0634
0635
0636
0637
0638

```

```

SUBROUTINE SQROOT(A,CC,G)
IMPLICIT REAL*8(A-H,O-Z),INTEGER(I-N)
DIMENSION A(7,6),E(7,6),C(7,6),AA(7,6),CC(7,6),DD(7,6)
COMMON NUMBER
FAC=G(2,1)
DO 5 I=2,7
DO 5 J=1,6
G(I,J)=1.DO/FAC*G(I,J)
CC(I,J)=1.DO/DSQRT(FAC)*CC(I,J)
5 CONTINUE
NUMBER=NUMBER+1
E(1,1)=NUMBER
E(1,2)=6.DO
E(1,3)=6.DO
NUMBER=NUMBER+1
DD(1,1)=NUMBER
DD(1,2)=6.DO
DD(1,3)=6.DO
DO 50 I=2,7
DO 50 J=1,6
IF(I-1-J)52,54,52
54 E(I,J)=DSQRT(G(I,J))
AA(I,J)=0.DO
DD(I,J)=CC(I,J)-E(I,J)
GO TO 50
52 E(I,J)=0.DO
AA(I,J)=G(I,J)
DD(I,J)=CC(I,J)
50 CONTINUE
DO 75 K=1,50CC
CALL ULTMA(G,DD,DD,7,6,6,7)
DO 60 I=2,7
DO 60 J=1,6
A(I,J)=(1.DO/(E(I,I-1)+E(J+1,J)))*(AA(I,J)-G(I,J))
60 CONTINUE
CALL ADDEM(CC,A,DD,-1.DO,7,6)
DO 65 I=2,7
DO 65 J=1,6
PT=DMAX1(CC(I,J))
65 CONTINUE
DO 70 I=2,7
DO 70 J=1,6
DD(I,J)=A(I,J)
70 CONTINUE
IF(DABS(PT)-1.D-10)80,80,75
75 CONTINUE
80 CONTINUE
CALL ADDEM(A,DD,E,1.DO,7,6)
DO 10 I=2,7
DO 10 J=1,6
A(I,J)=DSQRT(FAC)*A(I,J)
10 CONTINUE
RETURN
END

```

```

C SUBROUTINE SQUAM FORMS SQUARE MATRIX FROM LARGE COLUMN
SUBROUTINE SQUAM(A,B)
REAL*8 A,B
DIMENSION A(7,6),B(22)
COMMON NUMBER
NUMBER=NUMBER+1
N=0
NNN=1
80 N=N+1
IF(N-100)86,85,85
86 M=N*(N+1)/2
MM=B(1)+.1
IF(MM-M)80,81,80
85 WRITE(6,84)NUMBER,NNN
84 FORMAT(40H0 INTEGER ERROR IN SQUAM TO GET NUMBER ,2I7)
STOP
81 A(1,1)=NUMBER
A(1,2)=N
A(1,3)=N
NO=1
DO 82 J=1,N
DO 82 K=1,J
NO=NO+1
JJ=J+1
KK=K+1
A(JJ,K)=B(NO)
A(KK,J)=B(NO)
82 CONTINUE
87 IF(NO-1-MM)87,88,87
NNN=2
WRITE(6,84) NUMBER,NNN
88 CONTINUE
RETURN
END

```

0693
0694
0695
0696
0697
0698
0699
0700
0701
0702
0703
0704
0705
0706
0707
0708
0709
0710
0711
0712
0713
0714
0715
0716
0717
0718
0719
0720
0721
0722
0723
0724
0725
0726
0727

```

C SUBROUTINE TRACM COMPUTES TRACE OF SQUARE MATRIX
SUBROUTINE TRACM(T,A)
  REAL*8 A,T
  DIMENSION A(7,6)
  N=A(1,1)+.1
  IF(DABS(A(1,2)-A(1,3))-1.D-1)680,681,681
681 WRITE(6,682) N
682 FORMAT(49H NONSQUARE MATRIX ARGUMENT OF TRACM, MATRIX NO. , I7)
      STOP
680 M=A(1,2)+.1
      T=0.DO
      DO 683 J=1,M
        JP=J+1
        T=T+A(JP,J)
683 CONTINUE
      RETURN
      END

```

0728
0729
0730
0731
0732
0733
0734
0735
0736
0737
0738
0739
0740
0741
0742
0743
0744

```

C SUBROUTINE TRANM TRANSPPOSE MATRIX 7XN INTO (N+1)X6
SUBROUTINE TRANM(A,B,LL,L)
  REAL*8 A,B
  DIMENSION A(LL,6),B(7,L)
  COMMON NUMBER
  NUMBER=NUMBER+1
  A(1,1)=NUMBER
  A(1,2)=B(1,3)
  A(1,3)=B(1,2)
  M1=A(1,2)+.1
  N=A(1,3)+.1
  DO1019 J=2,M1
  DO1019 K=1,N
    KP=K+1
    JM=J-1
    A(J,K)=B(KP,JM)
1019 CONTINUE
      RETURN
      END

```

0745
0746
0747
0748
0749
0750
0751
0752
0753
0754
0755
0756
0757
0758
0759
0760
0761
0762
0763

```

C SUBROUTINE ULTMA MULTIPLY MATRICES A(N,M)=B(N,P)XC(PXM), WHERE N,M AND
C P ARE FROM 6 TO 50
SUBROUTINE ULTMA(A,B,C,MM,M,L,LL)
REAL*8 A,B,C,T
DIMENSION A(MM,M),B(MM,L),C(LL,M)
COMMON NUMBER
NUMBER=NUMBER+1
A(1,1)=NUMBER
IF(DABS(B(1,3)-C(1,2))- .1)1015,1012,1012
1012 WRITE(6,1013) NUMBER
1013 FORMAT(48H NONCCMFORMLABLE MATRIX MULTIDEE TO FORM NUMBER , I7)
3333 STOP
1015 A(1,2)=B(1,2)
A(1,3)=C(1,3)
M1=A(1,2)+1.1DO
N=A(1,3)+.1DC
NN=B(1,3)+.1DO
DO 1017 I=2,M1
DO 1017 J=1,N
T=0.00
DO 1018 K=1,NN
KK=K+1
T=T+B(I,K)*C(KK,J)
1018 CONTINUE
A(I,J)=T
1017 CONTINUE
RETURN
END

C SUBROUTINE ULTMB MULTIPLY MATRIX BY COLUMN
SUBROUTINE ULTMB(A,B,C,LL,L)
REAL*8 A,B,C,T
DIMENSION A(LL),B(LL,L),C(LL)
COMMON NUMBER
NUMBER=NUMBER+1
IF(DABS(B(1,3)-C(1,1))- .1DO)25,26,26
26 WRITE(6,27) NUMBER
27 FORMAT(38H NONCCMFORMLABLE ULTMB TO GET NUMBER ,I7)
STOP
25 A(1)=B(1,2)
M=B(1,2)+1.1DO
N=B(1,3)+.1DO
DO 28 I=2,M
T=0.00
DO 29 J=1,N
JJ=J+1
T=T+B(I,J)*C(JJ)
29 CONTINUE
A(I)=T
28 CONTINUE
RETURN
END

```



```

SUBROUTINE OUTPUT(R,B,U,V,T1,T2,T3,T5,LL,L,BETA)
IMPLICIT REAL*8(A-H,O-Z),INTEGER(I-N)
DIMENSION R(7,6),B(7,6),U(7,L),V(7,L),T1(7,L),T2(7,L),T3(LL,6),
1 T4(7,6),Q(5),T5(LL,6),D(7,6),E(7,6)
WRITE(6,32)
32 FORMAT(19H0 DESIRED MATRIX IS )
CALL PRIMM(R,6)
CALL ULTMA(T1,R,U,7,L,6,7)
CALL ADDEM(T2,T1,V,-1.00,7,L)
CALL TRANM(T3,T2,LL,L)
CALL ULTMA(T4,T2,T3,7,6,L,LL)
CALL TRACM(A,T4)
O(1)=A
CALL ULTMA(T2,B,V,7,L,6,7)
CALL ADDEM(T1,U,T2,-1.00,7,L)
CALL TRANM(T5,T1,LL,L)
CALL ULTMA(D,T1,T5,7,6,L,LL)
CALL TRACM(A,D)
O(2)=A
CALL ULTMA(E,T1,T3,7,6,L,LL)
CALL TRACM(A,E)
O(3)=A
CALL ADDEM(E,T4,D,BETA,7,6)
TETA=1.00+BETA
DO 35 I=2,7
DO 35 J=1,6
E(I,J)=E(I,J)/TETA
35 CONTINUE
CALL TRACM(A,E)
O(4)=A
WRITE(6,40)BETA
40 FORMAT(9H0 BETA = , D12.6 )
DO 30 I=1,4
30 WRITE(6,31)I,O(I)
31 FORMAT(20H0 CRITERION NUMBER ,I4,8H EQUALS ,1PE14.6)
END

```

APPENDIX C.

SAMPLE PROBLEM SOLVED BY COMPUTER

C-1 Sample Problem Using Program CECI 1 with 0% and 10% Maximum Random Errors.

The nomenclature used in the output of the program is as follows:

MODE = 1	Data will be generated in the program.
Matrix P	Original matrix C as read from cards.
Matrix BE	Matrix U as read from cards.
Matrix EB	Matrix V from $V = CU$.
Matrix U	Matrix BE with random errors introduced.
Matrix V	Matrix EB with random errors introduced.
Output No. 1	The desired compliance matrix which satisfies criterion one.
Output No. 2	The desired compliance matrix which satisfies criterion two.
Output No. 3A	The desired compliance matrix which satisfies criterion three, using the iterative procedure for taking the square root of the non-symmetric matrix VV^TUU^T .
Output No. 3B	The desired compliance matrix which satisfies criterion three, using the iterative procedure of taking the square root of the non-symmetric matrix UU^TVV^T .
Output No. 3C	The desired compliance matrix which satisfies criterion three. It is the average of outputs 3A and 3B.
Output No. 3D	The desired compliance matrix which satisfies criterion three, using the procedure that twice takes square roots of symmetric matrices.

The printed output for the problem with zero error appears on pages 77 to 82 inclusive; that for the problem with 10% maximum error appears on pages 83 to 88 inclusive.

1, 6 ROWS, 6 COLUMNS

MATRIX BE FOLLOWS

2, 6 ROWS, 24 COLUMNS

[illegible]

MATRIX EB FOLLOWS

3,			6 ROWS, 24 COLUMNS					
MATRIX NUMBER								
-0.8730C	03	0.8280D	03	-0.1710C	03	0.3420D	03	-0.4518D
-0.2880C	03	-0.7956D	03	-0.0	03	0.4590D	03	-0.5616D
-0.4500C	01	0.4644D	03	-0.6444D	03	0.3870D	02	-0.4050D
-0.1620D	03	0.8100D	03	-0.3600D	03	0.1566D	04	-0.9378D
-0.1777D	04	-0.4842D	03	0.1710D	04	0.2340D	03	-0.4500D
-0.9846D	03	0.6462D	03	-0.1179D	03	0.9990D	03	0.1478D
-0.2100D	03	0.1350D	03	0.1700D	03	0.9500D	02	0.4350D
-0.2550D	03	0.1750D	03	0.5500D	02	-0.2650D	03	-0.1600D
-0.3450D	03	0.2100D	03	0.1550D	03	0.7500D	02	0.3315D
-0.1680D	03	0.4080D	03	0.1800D	03	0.0	03	0.9336D
-0.7680D	03	0.4920D	03	-0.4920D	03	-0.2520D	03	0.3720D
-0.1920D	03	0.2400D	03	0.8040D	03	-0.6960D	03	0.9108D
-0.2000D	03	0.4480D	03	0.5520D	03	-0.2560D	03	0.6640D
0.1600D	02	-0.3120D	03	-0.1040D	03	0.4080D	03	-0.2000D
0.3536D	03	-0.6400D	01	-0.4400D	03	0.2680D	02	0.1296D
0.1520D	04	0.8640D	03	-0.6102D	03	0.1228D	04	0.1594D
-0.1809D	04	0.5670D	03	-0.1350D	03	-0.2160D	03	-0.5265D
-0.1323D	04	-0.7830D	03	0.1728D	04	0.2052D	04	

ERROR IN BLANCA = 0.0

ERROR IN BLANCA = 0.0

MATRIX U FOLLOWS

4,			6 ROWS, 24 COLUMNS					
MATRIX NUMBER								
-0.9700D	02	0.9200D	02	-0.1900D	02	0.3800D	02	0.5020D
-0.3200D	00	0.8840D	02	-0.0	02	0.5100D	02	-0.6240D
-0.5000D	00	0.5160D	02	-0.7160D	02	0.4300D	01	0.4500D
0.9000D	01	0.4500D	02	-0.2000D	02	0.8700D	02	-0.5210D
0.9870D	02	-0.2690D	02	-0.9500D	02	0.1300D	02	0.2500D
-0.5470D	02	0.3590D	02	-0.6550D	02	0.5550D	02	0.8210D
-0.4200D	02	-0.2700D	02	0.3400D	02	0.1900D	02	0.8700D
-0.5100D	02	0.3500D	02	0.1100D	02	-0.5300D	02	-0.3200D
-0.1400D	02	0.3400D	02	-0.1500D	02	0.0	02	0.6630D
-0.6400D	02	0.4100D	02	0.4100D	02	-0.2100D	02	0.3100D
-0.1600D	02	-0.2000D	02	0.6700D	02	-0.5800D	02	0.7590D
-0.2000D	01	-0.3900D	02	-0.6900D	02	0.5100D	02	0.8300D
0.2420D	02	0.3900D	02	-0.1300D	02	0.3350D	02	-0.2500D
0.5630D	02	0.3200D	02	-0.5500D	02	0.4550D	02	-0.1480D
-0.6700D	02	0.2100D	02	-0.2260D	02	-0.4550D	02	0.4800D
-0.4900D	02	-0.2900D	02	-0.5000D	01	-0.8000D	01	-0.2200D
						0.7600D	02	-0.1950D

MATRIX V FOLLOWS

MATRIX NUMBER		5, 6 ROWS, 24 COLUMNS					
-0.8730D 03	0.8280D 03	-0.1710D 03	0.3420D 03	0.2187D 03	0.6300D 01	-0.4518D 03	03
-0.2880D 03	-0.7956D 03	0.0	0.4590D 03	0.1080D 03	-0.2520D 03	-0.5616D 03	03
-0.4500D 01	0.4644D 03	0.3600D 03	0.3870D 04	0.2115D 03	-0.3303D 03	0.4050D 03	03
-0.1620D 03	0.8100D 03	-0.1710D 04	0.1566D 04	0.1890D 03	0.3744D 03	-0.9378D 03	03
-0.1777D 04	-0.4842D 03	0.1710D 04	0.2340D 03	0.1206D 04	0.3060D 03	-0.4500D 02	04
-0.9846D 03	0.6462D 03	-0.1179D 04	0.9990D 03	0.1352D 04	0.6210D 03	0.1478D 04	04
-0.2100D 03	0.1350D 03	0.1700D 03	0.9500D 02	0.3000D 03	-0.0	0.4350D 03	03
-0.2550D 03	0.2100D 03	0.5500D 02	-0.2650D 02	0.1200D 03	0.7500D 02	-0.1600D 03	03
-0.3450D 03	0.4080D 03	0.1550D 03	0.7500D 02	-0.2300D 02	0.1600D 02	0.3315D 03	03
-0.1680D 03	0.4920D 03	-0.1800D 03	0.0	0.6120D 03	-0.4680D 03	0.9360D 03	03
-0.7680D 03	-0.2400D 02	0.4920D 03	-0.2520D 03	0.4320D 03	0.7320D 03	0.3720D 03	03
-0.1920D 03	0.4480D 03	0.5520D 03	-0.6960D 03	-0.660D 03	0.7896D 03	0.9108D 03	03
-0.2000D 03	-0.3120D 03	-0.1040D 03	-0.2560D 03	0.0	-0.1040D 03	0.6640D 03	03
0.1600D 02	0.3120D 03	0.4400D 03	0.4080D 03	-0.1200D 03	-0.1680D 03	0.2000D 02	02
0.3536D 03	-0.6400D 01	0.6102D 03	0.2680D 03	-0.6880D 02	0.5048D 03	0.1184D 03	03
0.1520D 04	0.8640D 03	-0.6102D 03	0.1228D 04	-0.9450D 03	0.2403D 04	0.1296D 04	04
-0.1809D 04	0.5670D 03	-0.1350D 03	-0.2160D 03	-0.4050D 03	0.7830D 03	-0.5940D 03	03
-0.1323D 04	-0.7830D 03	0.1728D 04	0.2052D 04	-0.1021D 04	0.1031D 04	-0.5265D 03	03

ELAPSED TIME = 10.37

CUTPUT NO. 1, MINIMIZES V-SQUARE ERROR

DESIRED MATRIX IS

MATRIX NUMBER		17, 6 ROWS, 6 COLUMNS					
0.9000D 01	-0.1388D-15	-0.7355D-15	-0.8327D-16	-0.3886D-15	-0.2665D-14		
-0.1388D-15	0.1800D 02	0.3789D-14	0.1346D-14	-0.9437D-15	-0.1088D-13		
-0.7355D-15	0.3789D-14	0.5000D 01	-0.1069D-14	0.3053D-15	0.4885D-14		
-0.8327D-16	0.1346D-14	-0.1069D-14	0.1200D 02	0.1998D-14	0.8882D-15		
-0.3886D-15	-0.9437D-15	0.3053D-15	0.1998D-14	0.8000D 01	0.1044D-13		
-0.2665D-14	-0.1088D-13	0.4885D-14	0.8882D-15	0.1044D-13	0.2700D 02		
CRITERION NUMBER	1 EQUALS	1.797180D-22					
CRITERION NUMBER	2 EQUALS	3.555419D-25					
CRITERION NUMBER	3 EQUALS	4.837873D-24					
ELAPSED TIME =		1.48					

CUTPUT NC. 2, MINIMIZES U-SQUARE ERROR

DESIRED MATRIX IS

MATRIX NUMBER	33,	6 ROWS,	6 COLUMNS
0.9000	0.1	0.7640	0.15
0.7640	-0.15	0.1800	0.02
0.2049	-0.15	0.9709	0.15
0.1874	-0.15	0.6674	0.15
0.1027	-0.14	0.4137	0.15
0.1910	-0.14	0.1265	0.13
		0.2049	-0.15
		0.9709	-0.15
		0.5000	0.01
		0.5529	0.16
		0.1214	-0.14
		0.5123	-0.15
		0.2082	-0.15
		0.6147	-0.15
		0.9368	-0.15
		0.2700	0.02

CRITERION NUMBER	1	EQUALS	1.006397D-22
CRITERION NUMBER	2	EQUALS	7.975143D-25
CRITERION NUMBER	3	EQUALS	6.675377D-24

ELAPSED TIME = 0.96

OUTPUT NO.3A, MINIMIZES TRIANGLE ERROR

DESIRED MATRIX IS

MATRIX NUMBER	52,	6 ROWS, 6 COLUMNS			
0.900D-01	0.4025D-15	0.2234D-14	-0.9714D-15	-0.2665D-14	0.1110D-14
0.110D-13	0.1800D-02	0.1732D-13	-0.3331D-15	-0.9548D-14	-0.2198D-13
0.1908D-15	0.1360D-14	0.5000D-01	-0.3738D-14	-0.1416D-14	0.1110D-14
0.1527D-14	-0.6384D-15	0.3553D-14	0.1200D-02	-0.2442D-14	-0.3109D-15
0.1069D-14	-0.6661D-15	-0.3553D-14	0.1568D-14	0.8000D-01	-0.6661D-15
.0	-0.1155D-13	0.1021D-13	-0.2665D-14	0.1577D-13	0.2700D-02

CRITERION NUMBER	1	EQUALS	5.2452470-22
CRITERION NUMBER	2	EQUALS	8.4468380-25
CRITERION NUMBER	3	EQUALS	1.8336220-23

ELAPSEC TIME = 0.91

OUTPUT NO.38, MINIMIZES TRIANGLE ERROR

DESIRED MATRIX IS

MATRIX NUMBER	7C,	6 ROWS,	6 COLUMNS
0.9000D 01	0.1046D-13	0.3738D-15	0.1707D-14
0.8465D-15	0.1800D 02	0.1152D-14	-0.4163D-16
0.2012D-14	0.1754D-13	0.5000D 01	0.3331D-14
-0.3886D-15	-0.1305D-14	-0.3718D-14	0.1200D 02
-0.5107D-14	-0.1465D-13	-0.1429D-14	0.5551D-14
0.1110D-14	-0.1799D-13	0.1776D-14	-0.3331D-14
			0.2137D-14
			-0.8882D-15
			0.7327D-14
			-0.2220D-14
			0.1754D-13
			0.2700D 02

CRITERION NUMBER 1 EQUALS 6.412928D-22

CRITERION NUMBER 2 EQUALS 2.732927D-24

CRITERION NUMBER 3 EQUALS 3.17C094D-23

ELAPSED TIME = 0.87

OUTPUT NO.3C, MINIMIZES TRIANGLE ERROR

DESIRED MATRIX IS

MATRIX NUMBER	68,	6 ROWS,	6 COLUMNS
0.9000D 01	0.5433D-14	0.1304D-14	0.3678D-15
0.5926D-14	0.1800D 02	0.9236D-14	-0.1874D-15
0.1102D-14	0.9451D-14	0.5000D 01	-0.2038D-15
0.5690D-15	-0.9714D-15	-0.8283D-16	0.1200D 02
-0.2019D-14	-0.7661D-14	-0.2491D-14	0.3560D-14
0.5551D-15	-0.1477D-13	0.5995D-14	-0.2998D-14
			0.7661D-14
			0.8000D 01
			0.2637D-15
			-0.5218D-14
			-0.2151D-14
			0.1429D-14
			-0.2665D-14
			0.8438D-14
			0.2700D 02

CRITERION NUMBER 1 EQUALS 5.531635D-22

CRITERION NUMBER 2 EQUALS 1.355157D-24

CRITERION NUMBER 3 EQUALS 2.225371D-23

OUTPUT NC.3D, MINIMIZES TRIANGLE ERROR

DESIRED MATRIX IS

MATRIX NUMBER	109,	6 ROWS,	6 COLUMNS
0.8934D 01	0.1787D 00	-0.1065D 00	0.8814D-01
0.1787D 00	0.1804D 02	0.9723D 00	0.2940D 00
-0.1065D 00	0.9723D 00	0.4769D 01	0.8368D 00
0.8814D-01	0.2940D 00	0.8368D 00	0.1193D 02
0.1673D 00	-0.2398D 00	0.2083D-01	0.4856D 00
-0.3765D 00	0.4100D-01	0.9612D 00	0.9662D 00

CRITERION NUMBER 1 EQUALS 6.446506D 05

CRITERIGN NUMBER 2 EQUALS 8.064172D 03

CRITERIGN NUMBER 3 EQUALS 5.876357D 04

ELAPSED TIME = 1.89

TOTAL ELAPSED TIME = 16.58

MATRIX NUMBER 1, 6 ROWS, 6 COLUMNS

MATRIX RE FOLLOWS

MATRIX NUMBER 2, 6 ROWS, 24 COLUMNS

83

3, 6 ROWS, 24 COLUMNS

-0.87300	0.82800	-0.17100	0.34200	-0.89100	0.21870	0.3	-0.63000	0.1	-0.45180	0.3
-0.24800	-0.79560	-0.64440	0.45900	0.67500	0.10800	0.3	-0.25200	0.3	0.56160	0.3
-0.46200	0.46440	-0.36000	0.38700	0.13140	0.21150	0.3	-0.37000	0.3	0.40500	0.3
-0.81000	0.81000	-0.36000	0.15660	0.17460	0.18900	0.3	0.37440	0.3	0.93780	0.3
-0.17770	0.48420	-0.17100	0.23400	0.14260	0.12060	0.4	0.30600	0.3	0.45000	0.2
-0.98460	0.64620	-0.11790	0.99900	0.90000	0.13520	0.4	0.62100	0.3	0.14780	0.4
-0.21500	0.13500	-0.17000	0.95000	0.31000	0.17500	0.3	0.0	0.43500	0.3	
-0.34500	0.17500	-0.55000	0.26500	0.12000	0.30000	0.3	0.75000	0.2	0.16000	0.3
-0.16800	0.40800	-0.18000	0.75000	0.49000	0.23000	0.2	0.16000	0.2	0.33150	0.3
-0.76800	0.49200	-0.49200	0.25200	0.10200	0.61200	0.3	0.46800	0.3	0.93600	0.3
-0.24000	0.49200	-0.85000	0.69600	0.27600	0.43200	0.3	0.73200	0.3	0.37200	0.3
-0.31200	0.44000	-0.55200	0.25600	0.25800	0.66000	0.3	0.78960	0.3	0.91080	0.3
-0.64000	0.10400	-0.10400	0.48000	0.40000	0.0	0.3	0.10400	0.3	0.66400	0.3
-0.86400	0.44000	-0.44000	0.26800	0.88000	0.12000	0.3	-0.16300	0.3	0.20000	0.2
-0.56700	0.61000	-0.61000	0.12280	0.42400	0.68800	0.2	0.50480	0.3	0.11840	0.3
-0.78300	0.13500	-0.13500	0.21600	0.15010	0.94500	0.3	0.74300	0.4	0.12960	0.4
-0.78300	0.17280	-0.17280	0.20520	0.12690	0.40500	0.3	0.78300	0.3	0.59400	0.3
-0.21320	0.17280	-0.17280	0.20520	0.15360	0.10210	0.4	0.10310	0.4	0.52650	0.3

ERROR IN BLANCA = 0.01000

4, 6 ROWS, 24 COLUMNS

0.10520	0.20680	0.40620	0.93140	0.25310	0.76190	0.53230	0.02
-0.86580	-0.223180	0.02	-0.93140	0.2	0.76190	-0.53230	0.02
0.90870	0.77750	0.49270	0.69150	0.11110	0.29370	0.58920	0.02
-0.91990	-0.77750	0.49270	-0.69150	0.11110	-0.29370	-0.58920	0.02
0.40640	0.21220	0.90730	0.87440	0.24110	0.38500	0.42270	0.02
-0.40640	-0.21220	-0.90730	-0.87440	-0.24110	-0.38500	-0.42270	0.02
0.38260	0.10280	0.12310	0.85490	0.71400	0.16410	0.26750	0.02
-0.38260	-0.10280	-0.12310	-0.85490	-0.71400	-0.16410	-0.26750	0.02
0.59100	0.60280	0.51430	0.45280	0.78320	0.33860	0.77430	0.02
-0.59100	-0.60280	-0.51430	-0.45280	-0.78320	-0.33860	-0.77430	0.02
0.35240	0.34330	0.15800	0.67120	0.31570	0.20690	0.94590	0.02
-0.35240	-0.34330	-0.15800	-0.67120	-0.31570	-0.20690	-0.94590	0.02
0.42020	0.33090	0.56620	0.25050	0.60250	0.13680	0.31580	0.02
-0.42020	-0.33090	-0.56620	-0.25050	-0.60250	-0.13680	-0.31580	0.02
0.31990	0.16020	0.13650	0.94640	0.41790	0.29680	0.62910	0.02
-0.31990	-0.16020	-0.13650	-0.94640	-0.41790	-0.29680	-0.62910	0.02
0.41270	0.39460	0.56480	0.92110	0.52160	0.40080	0.76670	0.02
-0.41270	-0.39460	-0.56480	-0.92110	-0.52160	-0.40080	-0.76670	0.02
0.20940	0.67570	0.60690	0.19460	0.34650	0.66340	0.31240	0.02
-0.20940	-0.67570	-0.60690	-0.19460	-0.34650	-0.66340	-0.31240	0.02
0.51970	0.67790	0.33370	0.51150	0.40430	0.71130	0.77660	0.02
-0.51970	-0.67790	-0.33370	-0.51150	-0.40430	-0.71130	-0.77660	0.02
0.35110	0.13890	0.48270	0.10370	0.40430	0.22780	0.89640	0.02
-0.35110	-0.13890	-0.48270	-0.10370	-0.40430	-0.22780	-0.89640	0.02
0.83940	0.56510	0.32260	0.57410	0.13600	0.67150	0.24500	0.01
-0.83940	-0.56510	-0.32260	-0.57410	-0.13600	-0.67150	-0.24500	0.01
0.31510	0.20950	0.41260	0.57090	0.93300	0.89410	0.14270	0.02
-0.31510	-0.20950	-0.41260	-0.57090	-0.93300	-0.89410	-0.14270	0.02
0.29760	0.49130	0.83330	0.47950	0.15750	0.25210	0.22510	0.02
-0.29760	-0.49130	-0.83330	-0.47950	-0.15750	-0.25210	-0.22510	0.02
0.29640	0.61810	0.74520	0.61970	0.40120	0.39800	0.20710	0.02
-0.29640	-0.61810	-0.74520	-0.61970	-0.40120	-0.39800	-0.20710	0.02

OUTPUT NO. 2, MINIMIZES U-SQUARE ERROR

DESIRED MATRIX IS

MATRIX NUMBER	33,	6 ROWS,	6 COLUMNS
0.9357D C1	-0.1664D C0	-0.3592D-01	0.3113D C0
-0.1664D C0	0.1811D C2	0.8948D-01	-0.1677D C0
-0.3592D-01	0.8948D-01	0.4771D C1	0.1212D C0
0.3113D C0	-0.1677D C0	0.1212D C0	0.1178D C2
0.1078D C0	0.1174D C0	-0.4547D-01	-0.2360D C0
0.2026D C0	-0.1237D C0	0.5898D-01	-0.5429D-01
			0.1078D C0
			0.1174D C0
			-0.4547D-01
			-0.2360D C0
			-0.4939D-01
			0.2026D C0
			-0.1237D C0
			0.5898D-01
			0.2701D C2

CRITERION NUMBER 1 EQUALS 5.422693D 05

CRITERION NUMBER 2 EQUALS 2.133526D 03

CRITERION NUMBER 3 EQUALS 3.067654D 04

ELAPSED TIME = 1.65

OUTPUT NO.3A, MINIMIZES TRIANGLE ERROR

DESIRED MATRIX IS

MATRIX NUMBER	124,	6 ROWS,	6 COLUMNS
0.9312D C1	-0.1634D C0	-0.3204D-01	0.3155D C0
-0.1634D C0	0.1802D C2	0.1128D C0	-0.2446D C0
-0.3204D-01	0.1128D C0	0.4725D C1	0.1868D C0
0.3155D C0	-0.2446D C0	0.1868D C0	0.1174D C2
0.1280D C0	0.1366D C0	-0.6587D-01	-0.2198D C0
0.2673D-01	-0.1234D C0	0.1538D C0	-0.1347D C0
			0.1280D C0
			0.1366D C0
			-0.6587D-01
			-0.2198D C0
			-0.7312D-02
			0.2688D C2

CRITERION NUMBER 1 EQUALS 5.145427D 05

CRITERION NUMBER 2 EQUALS 2.193255D 03

CRITERION NUMBER 3 EQUALS 3.012705D 04

ELAPSED TIME = 1.61

MATRIX V FOLLOWS

MATRIX NUMBER		5, 6 ROWS, 24 COLUMNS	
-0.9106D	03	0.8613D	03
-0.2950D	03	-0.8119D	03
-0.4334D	01	0.4954D	03
0.1599D	03	-0.7295D	03
0.1850D	04	-0.4361D	03
-0.1011D	04	0.6908D	03
0.2075D	03	-0.1254D	03
-0.2365D	03	0.1902D	03
0.3318D	03	0.1937D	03
0.1586D	03	0.4164D	03
0.7717D	03	0.5217D	03
-0.1565D	03	-0.2517D	02
0.2135D	03	-0.4431D	03
0.1732D	02	-0.3389D	03
0.3215D	03	-0.6794D	01
0.1554D	04	0.7781D	03
-0.1657D	04	0.5720D	03
0.1225D	04	-0.8190D	03
ELAPSED TIME =		10.38	
OUTPUT NO. 1, MINIMIZES V-SQUARE ERROR			
DESIRED MATRIX IS			
MATRIX NUMBER		17, 6 ROWS, 24 COLUMNS	
C.9263D	01	-0.1383D	00
-0.1383D	00	0.1793D	02
-0.1683D	-01	0.1647D	00
0.3284D	00	0.4647D	01
0.1558D	00	0.2943D	00
-0.1802D	00	-0.1647D	00
CRITERION NUMBER		1 EQUALS 4.998192D 05	
CRITERION NUMBER		2 EQUALS 2.58462CD 03	
CRITERION NUMBER		3 EQUALS 3.145058D 04	
ELAPSED TIME =		1.47	

OUTPUT NO.3B, MINIMIZES TRIANGLE ERROR

DESIRED MATRIX IS

MATRIX NUMBER	214,	6 ROWS,	6 COLUMNS
C.9312D C1	-0.1634D C0	-0.3204D-C1	0.3155D C0
-0.1634D C0	C.1802D C2	0.1128D C0	-0.2446D C0
-0.3204D-C1	0.1128D C0	0.4725D C1	0.1868D C0
0.3155D C0	-0.2446D C0	0.1868D C0	0.1174D C2
0.128CD C0	0.1366D C0	-0.6587D-C1	-0.2198D C0
0.2673D-C1	-0.1234D C0	0.1538D C0	-0.1347D C0
			0.2688D C2

CRITERION NUMBER 1 EQUALS 5.145427D 05

CRITERION NUMBER 2 EQUALS 2.193255D 03

CRITERION NUMBER 3 EQUALS 3.012705D 04

ELAPSED TIME = 0.85

OUTPUT NO.3C, MINIMIZES TRIANGLE ERROR

DESIRED MATRIX IS

MATRIX NUMBER	212,	6 ROWS,	6 COLUMNS
C.9312D C1	-0.1634D C0	-0.3204D-C1	0.3155D C0
-0.1634D C0	C.1802D C2	0.1128D C0	-0.2446D C0
-0.3204D-C1	0.1128D C0	0.4725D C1	0.1868D C0
0.3155D C0	-0.2446D C0	0.1868D C0	0.1174D C2
0.128CD C0	0.1366D C0	-0.6587D-C1	-0.2198D C0
0.2673D-C1	-0.1234D C0	0.1538D C0	-0.1347D C0
			0.2688D C2

CRITERION NUMBER 1 EQUALS 5.145427D 05

CRITERION NUMBER 2 EQUALS 2.193255D 03

CRITERION NUMBER 3 EQUALS 3.012705D 04

OUTPUT NO.30, MINIMIZES TRIANGLE ERROR

DESIRED MATRIX IS

MATRIX NUMBER	253,	6 ROWS,	6 COLUMNS
0.9312D 01	-0.1634D 00	-0.3204D-C1	0.3155D 00
-0.1634D 00	0.1802D 02	0.1128D 00	-0.2446D 00
-0.3204D-01	0.1128D 00	0.4725D 01	0.1868D 00
0.3155D 00	-0.2446D 00	0.1868D 00	0.1174D 02
0.1280D 00	0.1366D 00	-0.6587D-01	-0.2198D 00
0.2673D-C1	-0.1234D 00	0.1538D 00	-0.1347D 00

CRITERION NUMBER 1 EQUALS 5.145427D 05

CRITERION NUMBER 2 EQUALS 2.193255D 03

CRITERION NUMBER 3 EQUALS 3.012705D 04

ELAPSED TIME = 1.99

TOTAL ELAPSED TIME = 18.06

C-2 Sample Problem Using Program CECI 2 with 0% and 10% Maximum Random Errors.

The nomenclature used in the output of the program is as follows:

MODE=1	Data will be generated in the program
BETA 0.8	(The sequence used is $\rho = .1, .2, \dots, .8$.)
Matrix P	Original matrix C as read from cards.
Matrix BE	Matrix U as read from cards.
Matrix EB	Matrix V from $V = CU$.
Matrix U	Matrix BE with random errors introduced.
Matrix V	Matrix EB with random errors introduced.
Output No. 4	The desired compliance matrix which satisfies criterion four.

The printed output for the problem with zero error appears on pages 90 to 93 inclusive; that for the problem with 10% maximum error appears on pages 94 to 97 inclusive.

MATRIX NUMBER 1, 6 ROWS, 6 COLUMNS

MATRIX BE FOLLOWS

MATRIX NUMBER 2, 6 ROWS, 24 COLUMNS

[illegible]

MATRIX ER FCLLCWS

MATRIX NUMBER			3,	6 ROWS, 24 COLUMNS			
-0.87300	03	0.82800	03	-0.17100	03	0.34200	03
-0.28800	03	-0.79560	03	-0.064440	03	0.45900	03
-0.45000	01	0.81000	03	-0.036000	03	0.38700	03
0.16200	03	-0.48420	03	-0.017100	04	0.15660	04
0.17770	04	-0.64620	03	-0.011750	04	0.23400	04
-0.98460	03	-0.13500	03	-0.011700	03	0.99900	03
-0.21000	03	0.17500	03	0.55000	02	0.95000	03
-0.34500	03	0.21000	03	0.15500	03	-0.26500	03
-0.16800	03	0.40800	03	-0.18000	03	0.75000	02
0.76800	03	0.49200	02	0.049200	03	0.0	03
-0.19200	03	-0.24000	03	0.080400	03	-0.25200	03
0.20000	03	-0.44800	03	0.55200	03	-0.69600	03
0.16000	02	-0.31200	03	-0.10400	03	0.25600	03
0.35360	04	-0.64000	01	0.44000	03	0.40800	03
0.15200	04	0.86400	03	-0.011020	03	0.26800	03
-0.18090	04	0.56700	03	-0.013500	03	0.02280	04
0.13230	04	-0.78300	03	-0.017280	04	-0.21600	03
						0.20520	04

ERROR IN BLANCA = 0.0

ERROR IN BLANCA = 0.0

MATRIX U FOLLOWS

MATRIX NUMBER			4,	6 ROWS, 24 COLUMNS			
-0.97000	02	0.92000	02	-0.19000	02	0.38000	02
-0.32000	00	0.51600	02	-0.071600	02	0.51000	02
-0.50000	01	0.45000	02	-0.020000	02	0.43000	01
0.98700	02	-0.26900	02	-0.095000	02	0.87000	02
-0.54700	02	-0.35900	02	-0.065500	02	0.13000	02
-0.42000	02	-0.27000	02	-0.034000	02	0.55500	02
-0.51000	02	0.35000	02	-0.011000	02	0.19000	02
-0.69000	02	0.42000	02	0.31000	02	-0.53000	02
-0.14000	02	0.34000	02	-0.015000	02	0.0	02
0.64000	02	0.41000	02	-0.041000	02	0.21000	02
-0.16000	02	-0.20000	01	0.67000	02	-0.58000	02
0.25000	01	-0.56000	02	-0.069000	02	-0.32000	02
0.20000	02	-0.39000	00	-0.13000	02	0.51000	02
0.44200	02	-0.32000	00	-0.055000	02	0.33500	02
0.56300	02	0.32000	02	-0.022600	02	0.45500	02
-0.67000	02	-0.21000	02	-0.050000	01	-0.80000	01
						0.76000	02

MATRIX NUMBER	5,	6 ROWS, 24 COLUMNS													
-0.87300	03	-0.17100	03	0.34200	03	0.45900	03	-0.89100	03	0.21870	03	0.63000	01	-0.45180	03
-0.28800	03	-0.04440	03	0.45900	03	0.38700	03	0.67500	03	0.10800	03	-0.25200	03	0.56160	03
-0.45000	01	-0.36000	03	0.38700	03	0.13140	03	0.13140	03	0.21150	03	-0.33030	03	0.40500	03
-0.16200	03	-0.17100	04	0.15660	04	0.17460	04	0.17460	04	0.18900	03	0.37440	03	-0.93780	03
-0.17770	04	-0.17100	04	0.23400	03	-0.14220	04	-0.14220	04	0.12060	04	0.30600	03	0.45000	02
-0.98460	03	-0.11790	03	0.99900	01	-0.09000	01	-0.09000	01	0.13520	04	0.62100	03	0.45000	04
-0.21000	03	-0.17000	02	0.95000	03	0.31000	03	0.31000	03	0.17500	03	-0.04350	03	0.14780	04
-0.13500	03	-0.55000	02	0.26500	03	0.12000	03	0.12000	03	0.30000	02	0.75000	02	0.16000	03
-0.34500	03	-0.15500	03	0.75000	02	0.49050	03	0.49050	03	-0.23000	02	0.16000	02	0.33150	03
-0.16800	03	-0.18000	03	0.00000	02	-0.10200	04	-0.10200	04	0.61200	03	-0.46800	03	0.93600	03
-0.76800	03	-0.49200	03	0.25200	03	-0.27600	03	-0.27600	03	0.43200	03	0.73200	03	0.37200	03
-0.19200	03	-0.80400	03	-0.69600	03	-0.25800	02	-0.25800	02	-0.66000	03	0.78960	03	0.91080	03
-0.24000	02	0.55200	03	-0.25600	03	0.40000	02	0.40000	02	0.00000	03	-0.10400	03	0.66400	03
-0.44800	03	-0.10400	03	0.40800	03	0.88000	02	0.88000	02	0.12000	03	0.16800	03	0.20000	02
-0.31200	03	-0.44000	03	0.26800	03	0.42400	02	0.42400	02	-0.68800	02	0.50480	03	0.11840	03
-0.86400	03	-0.61000	03	0.22800	04	-0.15010	04	-0.15010	04	-0.94500	03	0.24030	04	0.12960	04
-0.56700	03	-0.13500	03	0.12600	03	-0.12690	04	-0.12690	04	0.40500	03	0.78300	03	0.59400	03
-0.78300	03	-0.17280	04	0.20520	04	0.15360	04	0.15360	04	-0.10210	04	0.10310	04	-0.52650	03

ELAPSEC TIME = 0.90

16.01

OUTPUT NO.4, MINIMIZES V + BETA * U SQAPE ERRORS

DESIRED MATRIX IS

MATRIX NUMBER	17,	6	RCWS,	6	COLUMNS
C.90000 C1	-0.13880-15	-0.73550-15	-0.83270-16	-0.38860-15	-0.26650-14
-0.13880-15	C.18000 02	0.37890-14	0.13460-14	-0.94370-15	-0.10880-13
-0.73550-15	0.37890-14	0.50000 C1	-0.10690-14	0.30530-15	0.48850-14
-0.83270-16	0.13460-14	-0.10690-14	0.12000 02	0.19980-14	0.88820-15
-0.38860-15	-0.94370-15	0.30530-15	0.19980-14	0.80000 01	0.10440-13
-0.26650-14	-0.10880-13	0.48850-14	0.88820-15	0.10440-13	0.27000 02

BETA =0.800000 00

CRITERION NUMBER	1	EQUALS	1.797180D-22
CRITERION NUMBER	2	EQUALS	3.555419D-25
CRITERION NUMBER	3	EQUALS	4.837873D-24
CRITERION NUMBER	4	EQUALS	1.000014D-22

ELAPSED TIME = 0.99

TOTAL ELAPSED TIME = 12.86

MATRIX NUMBER 1, 6 ROWS, 6 COLUMNS

MATRIX NUMBER 1, 6 ROWS, 6 COLUMNS

[illegible]

MATRIX NUMBER 2, 6 RCWS, 24 COLUMNS

MATRIX NUMBER 2, 6 RCWS, 24 COLUMNS

97000	02	0.19000	02	0.38000	02	0.99000	02	0.24300	02	0.70000	00	-0.00000	02	0.50200	02
93200	00	-0.00000	02	0.51000	02	-0.75000	02	0.12000	02	0.28000	02	-0.00000	02	0.62400	02
90000	01	-0.71600	02	0.43000	01	0.14600	02	0.23500	02	-0.36700	02	-0.00000	02	0.45000	02
98700	02	-0.20000	02	0.87000	02	0.97000	02	0.10500	02	0.36000	02	-0.00000	02	0.52100	02
54700	02	-0.95000	02	0.13000	02	-0.79000	02	0.67000	02	0.17000	02	-0.00000	02	0.25000	01
42000	02	-0.65500	02	0.55500	02	-0.50000	00	0.75100	02	0.34500	02	-0.00000	02	0.82100	02
51000	02	-0.34000	02	0.19000	02	0.62000	02	0.35000	02	0.00000	02	-0.00000	02	0.87000	02
32000	02	-0.11000	02	0.53000	02	0.24000	02	0.60000	02	0.15000	02	-0.00000	02	0.32000	02
34000	02	-0.31000	02	-0.15000	02	-0.85000	02	-0.46000	01	0.32000	01	-0.00000	02	0.66300	02
41000	02	-0.15000	02	0.00000	02	0.85000	02	0.51000	02	0.39000	02	-0.00000	02	0.78000	02
67000	02	-0.41000	02	0.21000	02	-0.23000	02	-0.36000	02	0.61000	02	-0.00000	02	0.31000	02
69000	02	-0.69000	02	-0.58000	02	0.50000	01	-0.55000	02	0.58000	02	-0.00000	02	0.75900	02
39000	00	-0.13000	02	0.32000	02	0.50000	02	0.00000	02	0.13000	02	-0.00000	02	0.83000	02
30000	00	-0.13000	02	0.51000	02	0.11000	02	-0.15000	02	0.21000	02	-0.00000	02	0.25000	01
32000	00	-0.55000	02	0.33500	02	0.53000	01	-0.86000	01	0.63100	02	-0.00000	02	0.14800	02
26000	02	-0.55000	02	0.45500	02	0.55600	02	-0.35000	02	0.89000	02	-0.00000	02	0.48000	02
22000	02	-0.22000	02	0.80000	01	-0.47000	02	-0.15000	02	0.29000	02	-0.00000	02	0.22000	02
64000	02	-0.64000	02	-0.76000	02	0.56900	02	-0.37800	02	0.38200	02	-0.00000	02	0.19500	02

MATRIX NUMBER	3.	6 ROWS, 24 COLUMNS
-C-873CD	03	0.8289D
-C-2880D	03	-0.7956D
-C-450CD	C1	-0.4644D
-C-1620D	C3	-0.8100D
-C-1777D	C4	-0.4842D
-C-9846D	C3	-0.6462D
-C-21CCD	03	-0.1350D
-C-2450D	03	-0.1750D
-C-3450D	C3	-0.2100D
-C-168CD	C3	-0.4089D
-C-7680D	03	-0.4920D
-C-1920D	03	-0.2400D
-C-200CD	03	-0.4480D
-C-160CD	C2	-0.3120D
-C-3536D	03	-0.6400D
-C-1520D	04	-0.8640D
-C-1809D	C4	-0.5670D
-C-1323D	04	-0.7833D
-C-873CD	03	-0.8289D
-C-2880D	03	-0.7956D
-C-450CD	C1	-0.4644D
-C-1620D	C3	-0.8100D
-C-1777D	C4	-0.4842D
-C-9846D	C3	-0.6462D
-C-21CCD	03	-0.1350D
-C-2450D	03	-0.1750D
-C-3450D	C3	-0.2100D
-C-168CD	C3	-0.4089D
-C-7680D	03	-0.4920D
-C-1920D	03	-0.2400D
-C-200CD	03	-0.4480D
-C-160CD	C2	-0.3120D
-C-3536D	03	-0.6400D
-C-1520D	04	-0.8640D
-C-1809D	C4	-0.5670D
-C-1323D	04	-0.7833D
-C-873CD	03	0.3420D
-C-4590D	03	0.4590D
-C-3870D	C2	0.3870D
-C-1566D	04	0.1566D
-C-2340D	C3	0.2340D
-C-9990D	03	0.9990D
-C-9500D	C2	0.9500D
-C-7500D	C2	0.7500D
-C-4905D	03	0.4905D
-C-1020D	04	0.1020D
-C-2760D	03	0.2760D
-C-2580D	03	0.2580D
-C-4000D	C2	0.4000D
-C-8800D	C2	0.8800D
-C-4200D	C2	0.4200D
-C-1501D	04	0.1501D
-C-1269D	04	0.1269D
-C-1536D	04	0.1536D
-C-1021D	04	0.1021D
-C-2187D	03	0.2187D
-C-1080D	03	0.1080D
-C-2115D	03	0.2115D
-C-1890D	03	0.1890D
-C-1206D	04	0.1206D
-C-1352D	04	0.1352D
-C-1750D	03	0.1750D
-C-3000D	02	0.3000D
-C-2300D	02	0.2300D
-C-4320D	03	0.4320D
-C-6600D	03	0.6600D
-C-7896D	03	0.7896D
-C-1040D	03	0.1040D
-C-1680D	03	0.1680D
-C-5048D	03	0.5048D
-C-2403D	04	0.2403D
-C-7830D	03	0.7830D
-C-1031D	04	0.1031D
-C-6370D	01	0.6370D
-C-2520D	03	0.2520D
-C-3303D	03	0.3303D
-C-3744D	03	0.3744D
-C-3060D	03	0.3060D
-C-6210D	03	0.6210D
-C-4350D	03	0.4350D
-C-1600D	03	0.1600D
-C-3315D	03	0.3315D
-C-9360D	03	0.9360D
-C-3720D	03	0.3720D
-C-9108D	03	0.9108D
-C-6640D	03	0.6640D
-C-2000D	03	0.2000D
-C-1184D	03	0.1184D
-C-1296D	04	0.1296D
-C-5940D	03	0.5940D
-C-5265D	04	0.5265D

ERROR IN BLANCA = C.01000

MATRIX NUMBER	4,	6 ROWS, 24 COLUMNS
-0.10520	03	0.8658D
-0.29644D	02	-0.9087D
-0.51060	02	-0.5199D
-0.8827D	01	0.4064D
-0.9674D	02	-0.2810D
-0.5910D	02	0.3826D
-0.3815D	02	-0.2780D
-0.5325D	02	0.3524D
-0.6931D	02	0.4202D
-0.1522D	02	0.3199D
-0.6430D	02	0.4127D
-0.1679D	02	-0.2094D
-0.2273D	02	-0.5197D
-0.1853D	01	-0.3511D
-0.5499D	02	-0.8394D
-0.5346D	02	0.3151D
-0.6054D	02	0.2076D
-0.4815D	02	-0.2964D
-0.5323D	00	-0.2068D
-0.5892D	02	0.2318D
-0.4176D	02	-0.7779D
-0.5227D	02	-0.2122D
-0.2675D	02	0.1029D
-0.7745D	02	-0.6028D
-0.9459D	00	0.3433D
-0.3158D	02	0.9924D
-0.6291D	01	0.3309D
-0.7667D	02	0.1602D
-0.3126D	02	0.3946D
-0.4008D	02	-0.6757D
-0.6634D	02	0.6779D
-0.7113D	02	-0.1389D
-0.1226D	02	0.5651D
-0.2278D	02	-0.2095D
-0.6715D	01	-0.4913D
-0.8546D	02	0.6181D
-0.2921D	02	-0.6181D
-0.4251D	02	-0.2964D
-0.2251D	02	-0.2964D
-0.2071D	02	-0.2964D
-0.3980D	02	-0.2964D
-0.7619D	00	0.8658D
-0.2937D	02	-0.9087D
-0.3850D	02	-0.5199D
-0.2265D	02	0.4064D
-0.1641D	02	-0.2810D
-0.3386D	02	0.3826D
-0.2069D	00	-0.2780D
-0.1386D	02	0.3524D
-0.2968D	01	0.4202D
-0.4008D	02	0.3199D
-0.6634D	02	0.4127D
-0.7113D	02	-0.2094D
-0.1226D	02	-0.5197D
-0.2278D	02	-0.8394D
-0.6715D	01	-0.3511D
-0.8546D	02	-0.8394D
-0.2921D	02	0.3151D
-0.4251D	02	0.2076D
-0.2251D	02	-0.2964D
-0.2071D	02	-0.2964D
-0.3980D	02	-0.2964D
-0.7619D	00	0.8658D
-0.2937D	02	-0.9087D
-0.3850D	02	-0.5199D
-0.2265D	02	0.4064D
-0.1641D	02	-0.2810D
-0.3386D	02	0.3826D
-0.2069D	00	-0.2780D
-0.1386D	02	0.3524D
-0.2968D	01	0.4202D
-0.4008D	02	0.3199D
-0.6634D	02	0.4127D
-0.7113D	02	-0.2094D
-0.1226D	02	-0.5197D
-0.2278D	02	-0.8394D
-0.6715D	01	-0.3511D
-0.8546D	02	-0.8394D
-0.2921D	02	0.3151D
-0.4251D	02	0.2076D
-0.2251D	02	-0.2964D
-0.2071D	02	-0.2964D
-0.3980D	02	-0.2964D
-0.7619D	00	0.8658D
-0.2937D	02	-0.9087D
-0.3850D	02	-0.5199D
-0.2265D	02	0.4064D
-0.1641D	02	-0.2810D
-0.3386D	02	0.3826D
-0.2069D	00	-0.2780D
-0.1386D	02	0.3524D
-0.2968D	01	0.4202D
-0.4008D	02	0.3199D
-0.6634D	02	0.4127D
-0.7113D	02	-0.2094D
-0.1226D	02	-0.5197D
-0.2278D	02	-0.8394D
-0.6715D	01	-0.3511D
-0.8546D	02	-0.8394D
-0.2921D	02	0.3151D
-0.4251D	02	0.2076D
-0.2251D	02	-0.2964D
-0.2071D	02	-0.2964D
-0.3980D	02	-0.2964D
-0.7619D	00	0.8658D
-0.2937D	02	-0.9087D
-0.3850D	02	-0.5199D
-0.2265D	02	0.4064D
-0.1641D	02	-0.2810D
-0.3386D	02	0.3826D
-0.2069D	00	-0.2780D
-0.1386D	02	0.3524D
-0.2968		

MATRIX V FOLLOWS

MATRIX NUMBER	5,	6	ROWS, 24	COLUMNS
-0.9166D C3	-0.8613D C3	-0.1583D C3	0.3649D C3	0.6178D C1
-0.2950D C3	-0.8119D C3	-0.8306D C3	0.4871D C3	-0.2535D C3
-0.4334D C1	-0.4954D C3	-0.6116D C3	0.4212D C3	-0.3194D C3
-0.1599D C3	-0.7295D C3	-0.3537D C3	0.1674D C4	-0.3467D C3
-0.1850D C4	-0.4361D C3	-0.1644D C4	0.2109D C3	0.3021D C3
-0.1011D C4	-0.6908D C3	-0.1109D C3	0.1085D C4	0.5754D C3
-0.2075D C3	-0.1254D C3	-0.1608D C3	0.9320D C3	-0.9025D C0
-0.2565D C3	-0.1902D C3	-0.4965D C2	0.2491D C3	-0.8139D C2
-0.3318D C3	-0.1937D C3	-0.1681D C3	0.7065D C2	0.1606D C2
-0.1586D C3	-0.4164D C3	-0.1665D C3	0.8025D C3	-0.4714D C3
-0.7717D C3	-0.5217D C3	-0.4755D C3	-0.2292D C3	-0.3519D C3
-0.1965D C3	-0.2517D C2	-0.7592D C3	-0.7166D C3	0.9405D C3
-0.2135D C3	-0.4431D C3	-0.4986D C3	0.2510D C3	0.3500D C3
-0.1732D C2	-0.3389D C3	-0.1068D C3	0.3769D C3	0.9916D C3
-0.3215D C3	-0.6794D C1	-0.3982D C3	0.2751D C3	-0.6023D C3
-0.1554D C4	-0.7781D C3	-0.5619D C3	0.1262D C4	-0.2171D C2
-0.1857D C4	-0.5720D C3	-0.1464D C3	0.2086D C3	-0.1290D C3
-0.1225D C4	-0.8190D C3	-0.1702D C4	0.1861D C4	-0.1223D C4
				-0.6110D C3
				-0.4759D C3

. ELAPSED TIME = 10.95

. ELAPSED TIME = 1.86

NUMBER OF INTERACTIONS = 0.17000000 C2

DESIRED MATRIX IS

MATRIX NUMBER	17, 6 ROWS, 6 COLUMNS
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1
21	1
22	1
23	1
24	1
25	1
26	1
27	1
28	1
29	1
30	1
31	1
32	1
33	1
34	1
35	1
36	1
37	1
38	1
39	1
40	1
41	1
42	1
43	1
44	1
45	1
46	1
47	1
48	1
49	1
50	1
51	1
52	1
53	1
54	1
55	1
56	1
57	1
58	1
59	1
60	1
61	1
62	1
63	1
64	1
65	1
66	1
67	1
68	1
69	1
70	1
71	1
72	1
73	1
74	1
75	1
76	1
77	1
78	1
79	1
80	1
81	1
82	1
83	1
84	1
85	1
86	1
87	1
88	1
89	1
90	1
91	1
92	1
93	1
94	1
95	1
96	1
97	1
98	1
99	1
100	1

0.92640	0.13890	0.17160	0.32800	0.15550	0.0.17940
-0.13890	-0.17930	-0.16320	-0.33520	0.0.12880	00
-0.17160	-0.13320	-0.46510	0.0.28990	-0.94200	0.0.37050
0.32800	0.33520	0.28990	0.11700	-0.20300	0.0.23310
0.15550	0.16420	0.0.94200	0.0.20300	0.0.75750	0.0.33440
0.17940	0.12880	0.37050	0.23310	0.33440	0.26710

BETA = 0.800000D 00

CRITERION NUMBER	1	EQUALS	4.998249D 05
CRITERION NUMBER	2	EQUALS	2.57C142D 03
CRITERION NUMBER	3	EQUALS	3.135593D 04
CRITERION NUMBER	4	EQUALS	2.788228D 05

ELAPSED TIME = 0.99

TOTAL ELAPSED TIME = 13.85

APPENDIX D

BRIEF TREATMENT OF RELATED PROBLEMS

D-1 Case where there are no Restrictions on C.

In the development of this thesis, starting with the matrix equation $CU = V$, matrix C was restricted to be symmetric and positive definite. Suppose now that these restrictions should be removed, and, indeed, let it only be required that C have the correct dimensions to relate matrices U and V which may have a different number of rows.

Let us write equation (4) in a modified form

$$\alpha = \text{tr} [(CU - V)^T Z (CU - V)] \quad (\text{D-1})$$

where Z is arbitrary and might be thought of as representing a set of weighting factors. If Z is the unit matrix, equation (D-1) reduces to equation (4).

From equation (D-1)

$$\alpha = \text{tr} [U^T C^T Z CU - U^T C^T Z V - V^T Z CU + V^T Z V]$$

Note that, in general, $C^T \neq C$ since the requirement of symmetry is not made here.

Using * to denote partial differentiation with respect to any element x, as before, one gets

$$\begin{aligned} \alpha^* &= \text{tr} [U^T C^T * Z CU + U^T C^T Z C^* U - U^T C^T * Z V - V^T Z C^* U] \\ &= \text{tr} [(CUU^T - VU^T) (C^T *) (Z + Z^T)] \end{aligned} \quad (\text{D-2})$$

Thus α is made stationary with respect to each element of C if C satisfies the equation

$$CUU^T = VU^T$$

If U is of rank m, where m is the number of rows in U, then

$$C = VU^T(UU^T)^{-1}$$

It is interesting to note, that the result is independent of the arbitrary weighting matrix Z.

D-2 Case where C is Orthogonal.

This problem arises in the treatment of the orientation of a rigid body.

If C is orthogonal, then

$$C^T = C^{-1} = Y$$

where the notation $Y = C^{-1}$ is used for convenience in manipulation.

As before let x be any element of C, and let * denote $\frac{\partial}{\partial x}$. Then the relation $CY = I$, leads to $C^* = -CY^*C$, and equation (4) can be written as

$$\alpha = \text{tr} (U^T U - U^T Y V - V^T C U + V^T V)$$

so that

$$\begin{aligned} \alpha^* &= \text{tr} (-U^T Y^* V + V^T C Y^* C U) \\ &= \text{tr} [(Y^*) (C U V^T C - V U^T)] \end{aligned} \quad (D-3)$$

Thus, α is stationary with respect to variation in any of the elements of C, if C satisfies the equation

$$CUV^T C = VU^T$$

which will be so if C is any of the several values

$$C = (VU^T U V^T)^{\frac{1}{2}} (U V^T)^{-1}$$

VU^TUV^T is symmetric matrix, and its square root can be obtained as described in Section 4.

These results were obtained by Professor Brock and are given here so as to collect in one place information on this general matrix problem.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2. Library Naval Postgraduate School Monterey, Calif. 93940	2
3. Professor John E. Brock Mechanical Engineering Department Naval Postgraduate School Monterey, Calif. 93940	3
4. Department of Mechanical Engineering Naval Postgraduate School Monterey, Calif. 93940	1
5. Mr. Carl P. Wilson Code 452 General Dynamics/Electric Boat Division Eastern Point Road Groton, Conn. 03640	3
6. Mr. Leslie Kaldor Navy Marine Engineering Laboratory Annapolis, Maryland 21402	3
7. LT Cesar F. Villaran Tapia, Peruvian Navy Moreyra y Riglos 769, Pueblo Libre Lima, Peru. S.A.	4
8. Commander Naval Ship Systems Command Navy Department Washington, D. C. 20360	1

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Naval Postgraduate School Monterey, California 93940		2a. REPORT SECURITY CLASSIFICATION Unclassified	
		2b. GROUP	
3. REPORT TITLE Determination of Optimal Compliance and Stiffness Matrices from Experimental Data			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) None			
5. AUTHOR(S) (Last name, first name, initial) VILLALBA TAPI, Cesar E., LT., Peruvian Navy			
6. REPORT DATE September 1967		7a. TOTAL NO. OF PAGES 102	7b. NO. OF REFS 7
8a. CONTRACT OR GRANT NO.		9a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO. N/A		N/A	
c. N/A		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.		N/A	
10. AVAILABILITY/LIMITATION NOTICES Information contained herein is classified "Secret" and is to be controlled in accordance with the provisions of the Naval Postgraduate School's security policy.			
11. SUPPLEMENTARY NOTES N/A		12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School	

13. ABSTRACT

A linear structure can be characterized by its compliance matrix C , which is 6×6 symmetrical and positive definite and which relates a force 6-vector \bar{F}_j to a displacement 6-vector \bar{D}_j by the relation $C\bar{F}_j = \bar{D}_j$. The inverse, S , of C , is called the stiffness matrix and satisfies $S\bar{D}_j = \bar{F}_j$. This thesis deals with the problem of finding optimal values of such matrices C and S from experimental determinations of a sufficient number of vector-pairs (\bar{F}_j, \bar{D}_j) which are presumed to contain random errors.

J. E. Brock has introduced this problem area, suggested several different criteria of optimality, and solved some of the corresponding specific problems. This thesis completes the solution to a previously unsolved specific problem of this group and contributes computationally convenient new solutions to another. Moreover, a computer program, originally written for the CDC 1604 has been rewritten, in FORTRAN IV Language, as two programs for the IBM System 360 computer, and the capability has been significantly augmented.

14.

KEY WORDS

LINK A

LINK B

LINK C

ROLE

WT

ROLE

WT

ROLE

WT

Matrices
Compliance
Stiffness
Analysis of Data
Experimental Data
Data Reduction
Linear Systems
Least-Squares
Experiment
Optimal Matrices
Matrix Equations
Matrix Square Roots

—

thesV68

DUDLEY KNOX LIBRARY



3 2768 00414859 3

DUDLEY KNOX LIBRARY